

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Новгородский государственный университет имени Ярослава Мудрого»  
Институт электронных и информационных систем

---

Кафедра «Информационных технологий и систем»

Допустить к защите  
Заведующий кафедрой ИТиС

\_\_\_\_\_ Л.Н. Цымбалюк  
«\_\_\_» \_\_\_\_\_ 2024г.

Автоматизированная система для работы ГЭК  
Пояснительная записка к выпускной квалификационной работе  
по направлению 09.03.01 «Информатика и вычислительная техника»,  
профиль «Программное обеспечение вычислительной техники и  
автоматизированных систем»

НУОП.90001-01 81

Руководитель  
\_\_\_\_\_ Соколова Г. Ю.  
«\_\_\_» \_\_\_\_\_ 2024 г.

Студент группы 0091  
\_\_\_\_\_ Складник Л. С.  
«\_\_\_» \_\_\_\_\_ 2024 г.

Великий Новгород, 2024 г.

## **Аннотация**

Выпускная квалификационная работа на тему: «Автоматизированная система работы ГЭК».

Объём работы:

страниц	100
рисунков	25
таблиц	08
приложений	1
библиографических источников	22

В рамках этой выпускной квалификационной работы была разработана автоматизированная система, которая должна улучшить работу для ГЭК.

Выпускная квалификационная работа состоит из двух глав. Первая - анализ предметной области, вторая - разработка и реализация проектных решений.

Разработанная автоматизированная система удовлетворяет всем требованиям технического задания.

### **Annotation**

Final qualification work on the topic: "Automated system of work of the GEK".

Volume of work:

Pages 100

figures 25

tables 08

appendices 1

bibliographic sources 22

As part of this final qualification work, an automated system was developed that should improve the work for the GEK.

The final qualifying work consists of two chapters. The first is the analysis of the subject area, the second is the development and implementation of design solutions.

The developed automated system meets all the requirements of the terms of reference.

## Содержание

Введение .....	7
1. Раздел 1. Теоретическая составляющая работы.....	10
1.1. Анализ предметной области и выявление прототипов.....	10
1.2. Проектирование .....	11
1.2.1. Выбор языка программирования .....	11
1.2.2. Выбор среды разработки .....	12
1.2.3. Выбор библиотек и фреймворков.....	13
1.2.4. Выбор сервера.....	15
1.2.5. Выбор архитектуры.....	16
1.2.6. Выбор СУБД.....	16
1.3. Анализ требований технического задания.....	17
1.3.1. Функциональные требования.....	17
1.3.2. Требования к интерфейсу.....	18
1.3.3. Требования к реализации .....	18
1.3.4. Требования к надежности .....	19
1.3.6. Требования к установке .....	20
1.3.7. Требования к безопасности .....	20
2. Раздел 2. Практическая составляющая работы .....	21
2.1. Реализация .....	21
2.1.1. Реализация базы данных .....	21
2.1.2. Физическое моделирование АИС .....	25
2.1.3. Реализация интерфейса .....	36
2.2.1. Набор тестов.....	43
Заключение .....	46
Список источников .....	47
Приложение А.....	49

## Перечень сокращений и специальных терминов

**Адаптив** — адаптивный дизайн, то есть дизайн веб-страниц, обеспечивающий правильное отображение сайта на разных устройствах.

**База данных** - представляет собой совокупность связанных таблиц, предназначенных для хранения определенной информации в рамках проекта.

**ВКР** – выпускная квалификационная работа

**ГЭК** – государственная экзаменационная комиссия

**Информационная система** – это приложение, предназначенное для хранения данных и их обработки. Основой информационной системы является база данных с информацией, хранящейся в одной или нескольких связанных между собой таблицах.

**ПЗ** – Пояснительная записка

**ПП** – Программный Продукт

**Реляционная модель** – это модель данных, которая помогает обеспечить согласованность данных в системе баз данных. В основе модели лежит концепция таблиц и их отношений, где каждая таблица представляет собой набор связанных данных, а каждая строка в таблице - отдельную запись или экземпляр этих данных. Каждый столбец в таблице представляет определенный атрибут или поле данных.

**ТЗ** – техническое задание

**SQL** — это язык программирования, который позволяет делать запросы к данным, фильтровать и сортировать информацию.

**Express** — это простейшая, гибкая и оптимизированная платформа Node.js, упрощающая разработку веб-приложения, которое может обрабатывать различные типы запросов, например GET, PUT, POST и delete.

**Node.js** - кроссплатформенная среда исполнения с открытым исходным кодом, которая позволяет разработчикам создавать всевозможные серверные инструменты и приложения используя язык JavaScript.[1]

**npm** - это менеджер пакетов, который управляет модулями и зависимостями проекта.

**JavaScript** — это интерпретируемый язык программирования, который используют для написания frontend- и backend-частей сайтов, а также мобильных приложений.[1]

## **Введение**

Система высшего образования в России, адаптируясь к новым рыночным отношениям, переживает сложные преобразования. В условиях изменения потребностей в профессиях современное общество предъявляет достаточно серьезные требования к специалисту. Цели современного образования меняются и акцент переносится на формирование компетентности. В связи с введением федерального государственного образовательного стандарта (ФГОС) третьего поколения произошли изменения порядка оценки качества обучения. Данный стандарт позволяет максимально приблизить подготовку специалистов к условиям современного рынка труда, так как целью введения ФГОС третьего поколения явилось приближение образовательных услуг к реальным требованиям работодателей. Теперь именно работодатели определяют направление работы учебного заведения, начиная от рецензирования программ профессиональных модулей до обязательного участия в проверке освоения профессиональных и общих компетенций.

Формирование компетентной личности — это одна из главных задач высшего профессионального образования и заключается в формировании компетентной личности, способной к саморазвитию, самообразованию, инновационной деятельности. В современных условиях студенты и выпускники из пассивных потребителей знаний должны перейти в разряд активных специалистов, умеющих формулировать проблему, анализировать пути ее решения, искать способы разрешения проблемы и доказывать ее правильность.

Кроме задачи по формированию компетенций будущего специалиста, перед образовательным учреждением стоит проблема оценивания качества результатов обучения, сформированности компетенций обучающего и актуализирует вопрос об их оценивании.

Оценка качества подготовки выпускников в новой компетентностно-методологической парадигме профессионального образования требует серьезного подхода к оцениванию приобретаемых обучающимися характеристик, формирующих их компетенции.

Государственная итоговая аттестация (ГИА) выпускников высшего профессионального образования, является обязательной и осуществляется в соответствии со следующими нормативно-правовыми документами:

1. Федеральный Закон от 29.12.2012 № 273-ФЗ «Об образовании в Российской Федерации».

2. Федеральный образовательный стандарт высшего профессионального образования по специальности.

3. Приказ Министерства образования и науки Российской Федерации от 5 апреля 2017 г. Приказ Министерства образования и науки Российской Федерации от 05.04.2017 №301 «Об утверждении Порядка организации и осуществления образовательной деятельности по образовательным программам высшего профессионального образования - программам бакалавриата, программам специалитета, программам магистратуры»;

Целью государственной итоговой аттестации является установление соответствия уровня и качества подготовки выпускников федеральным государственным образовательным стандартам высшего профессионального образования в части государственных требований к минимуму содержания и уровню подготовки выпускников, и дополнительным требованиям по конкретным специальностям, а так же готовности выпускника к самостоятельной профессиональной деятельности.

Процедуры подготовки и проведения государственной итоговой аттестации зависят от вида аттестации. Общие условия для проведения всех видов итоговой аттестации является организация и работа государственной аттестационной комиссии. В связи с тем, что по результатам работы комиссии составляется огромное количество документации, выявлена необходимость в



автоматизации процесса составления документации государственной итоговой аттестации.

Объектом выпускной квалификационной работы является ГИА, а предметом — документационное сопровождение процесса ГИА, а именно защиты ВКР.[19]

Цель данной работы — разработка автоматизированной системы работы ГЭК кафедры ИТИС, Института электронных и информационных систем, Новгородского государственного университета им. Ярослава Мудрого.

Предполагается, что после создания данного продукта, работа секретаря экзаменационной комиссии будет автоматизирована, сократится время, затраченное на формирование пакета документов и снизится вероятность порчи бланка оценочного листа.

Задачи, для решения поставленной цели:

- Проанализировать предметную область;
- Определить базовые требования к работе
- Разработать техническое задание по базовым требованиям;
- Спроектировать систему;
- Реализовать автоматизированную систему в программном коде;
- Протестировать ПП, исправить дефекты;
- Разработать пояснительную записку;

Поэтому в рамках данной пояснительной записки рассматривается разработка автоматизированной системы работы ГЭК. Данный проект позволит закрыть потребность в системе, обеспечивающей хранение результатов и работ и оценивания студентов

## **1. Раздел 1. Теоретическая составляющая работы**

### **1.1. Анализ предметной области и выявление прототипов**

Выпускная квалификационная работа является государственной аттестационной работой и формой итогового контроля обучения студентов.

Основная цель ВКР — выполнить реальный проект разработки программного продукта с соблюдением всех технологических процедур, продемонстрировать готовность выпускника самостоятельно решать инженерные задачи.

Итоговую оценку за ВКР формирует ГЭК после завершения процедуры защиты ВКР студентом.

Государственная экзаменационная комиссия представляет собой собрание экспертов, комиссию, которая призвана осуществлять итоговую аттестацию выпускников ВУЗов.

С течением времени происходит накопление большого количества документации, связанной с оцениванием ВКР по каждому студенту и члену ГЭК. Технологическое решение этой проблемы лежит в автоматизированной системе работы ГЭК. Потому что хранение ПЗ, ТЗ каждой ВКР, протоколы оценивания на каждого студента каждым членом ГЭК, результаты оценивания работ ГЭК, итоговые оценки в одном месте(продукте) очень удобно.

Так же в этой автоматизированной системе будет происходить сам процесс оценивания работ комиссией, выставления итоговой оценки, эти данные будут так же храниться в этой системе. То есть, в конечном итоге, должен быть продукт облегчающий весь процесс оценивания ВКР на кафедре ИТИС, института электронных и информационных систем, Новгородского государственного университета им. Ярослава Мудрого.

Но в ходе моей исследовательской работы было найдено несколько аналогов автоматизированных систем. Одна из наиболее подходящей для

рассмотрения была работа студента Российского государственного профессионально-педагогического университета «Программный продукт для протоколирования результатов государственной итоговой». Там был настроен весь рабочий процесс оценивания ГИА секретарем, председателем и членами ГЭК через 1С. На данный момент подобных систем крайне мало, и адаптированы только под свои ВУЗы, институты или даже, как в нашем случае, кафедры. Ведь система оценивания ВКР одна, но в разных учебных заведениях есть дополнительная документация, где происходят различные допустимые изменения оценивания.

Следствием данной проблемы будет отсутствие возможности сравнить результаты работы с любой другой автоматизированной системой других учебных учреждений.

### **Цели и задачи разработки**

Целью работы является разработка автоматизированной системы работы ГЭК. Назначение которого - хранение и предоставление данных с ВКР студентов, оценивание и создание итоговых протоколов.

В процессе разработки будет создана система, состоящая из:

- 1) веб-приложение, необходимое для работы ГЭК
- 2) базы данных, в которой хранится необходимая информация о студентах, их работы и оценки за ВКР
- 3) инструментов, которые предоставляют интерфейс взаимодействия с системой для работы.

## **1.2. Проектирование**

### **1.2.1. Выбор языка программирования**

Существует огромное множество языков и платформ, позволяющих создать полноценный сервер, и нельзя сказать наверняка какой из них лучше. Наиболее популярные языки для написания серверных приложений это PHP, JavaScript, Python, Ruby, Java, C++, Perl и множество других.

Выбор инструментов разработки главным образом был обусловлен получением нового опыта разработки на языке программирования JavaScript.

С момента появления язык JavaScript прошёл долгий путь от браузерного языка до одного из важнейших языков во всех областях разработки программного обеспечения, на что существенно повлияло появление таких технологий, как Node.js, React.js и Electron. Именно Node.js был выбран в качестве веб-сервера Journal System.

Node.js - чрезвычайно популярная программная платформа, которая позволяет использовать JavaScript для простого создания масштабируемых серверных приложений. Это позволяет создавать эффективный код, обеспечивая более устойчивый способ написания программного обеспечения только на одном языке во всем стеке, наряду с предельными уровнями повторного использования, прагматизма, простоты и совместной работы. Node.js революционизирует веб и то, как люди и компании создают свое программное обеспечение.[13]

Одной из сильных сторон Node.js является однопоточная модель программирования: Node.js работает с моделью программных потоков, используемой в браузере. На стороне сервера подобный подход возможен благодаря трём концепциям Node.js: событиям, асинхронным API и неблокирующему вводу/выводу.

Исходя из вышеперечисленного, для разработки автоматизированной системы для работы ГЭК был выбран язык программирования JavaScript.

### **1.2.2. Выбор среды разработки**

Для создания простых программ на языке Java Script достаточно обычного текстового редактора, например – Sublime Text, однако, для увеличения скорости и удобочитаемости кода рекомендуется использовать специализированные программы. Наиболее популярными из таких программ являются редакторы кода, такие как: Microsoft Visual Studio Code, WebStorm, JetBrains Rider, Sublime Text, Eclipse. Все они схожи по функционалу и представляют собой

усовершенствованный блокнот: имеется подсветка синтаксиса, авто-табуляция, возможность сворачивания блоков кода, авто дополнение и многое другое.

Однако для создания полноценного сервера на языке Java Script редакторов кода может быть недостаточно. Что бы разработка программы была максимальна удобна необходимо использовать специализированные среды разработки.

Для разработки системы была выбрана среда разработки – WebStorm, позиционирует себя как «лёгкий» редактор кода для разработки веб-приложений, включает в себя авто завершение и инспекции кода, подсветка синтаксиса и ошибок, исправления, отладка, система контроля версий и рефакторинг, встроенные инструменты для работы с БД, а потому и выбрана для данной выпускной квалификационной работы. Данная среда разработки предлагает весь функционал описанных выше редакторов кода, а также автоматическую установку модулей, авто-дополнение методов из подключаемых модулей с описанием входных и выходных данных.

### **1.2.3. Выбор библиотек и фреймворков**

В качестве фреймворка был выбран Express.

Express – «минималистичный и гибкий фреймворк для Node.js- веб-приложений, обеспечивающий набор возможностей для построения одно- и многостраничных и гибридных веб-приложений».[13]

Сильной стороной Express является его минималистичность и гибкость.

Не существует «правильного способа» построения веб-приложения используя Express: программист сам волен определяться с архитектурой

Body-parser — это пакет, который позволяет разбирать тела запросов, хранящиеся в req.body, выступая в роли промежуточного слоя для серверов, основанных на Express. Он обрабатывает запросы до их попадания в соответствующие обработчики.

Cors — это пакет, содержащий реализацию ПО промежуточного слоя для Connect/Express, которое позволяет настраивать и использовать CORS.

Morgan — промежуточное программное обеспечение регистратора HTTP-запросов для node.js.

Http-errors — это библиотека, позволяющая генерировать HTTP-ошибки.

Config- библиотека, которая позволяет создавать и управлять файлами конфигурации в приложении Node для разных сред развёртывания.

Crypto - это один из сторонних модулей, которые помогают шифровать, расшифровывать или хэшировать любые данные.

Dotenv - модуль с открытым исходным кодом, который упрощает управление переменными среды, инструментами и настройками в приложениях Node.js.

Puppeteer- библиотека, которая позволяет настроить работу с браузером Google Chrome. В частности, с помощью этой библиотеки можно создавать программы для автоматического сбора данных с веб-сайтов.

Knex - библиотека, выполняющая функции построителя запросов. Он позволяет строить SQL запросы с использованием функций JavaScript и точечной нотации.

Nodemon- это инструмент, который помогает разрабатывать приложения на основе Node.js. Он автоматически перезапускает приложение при обнаружении изменений файлов в каталоге.

Passport - самая популярная библиотека для аутентификации в node.js, хорошо известная сообществу и успешно используемая во многих production приложениях. objection

Hbs- шабланизатор Node.js

serve-static — библиотека обслуживающая статический контент, подобный одностраничному приложению .

jsonwebtoken - это библиотека для создания (подписания) и

подтверждения (проверки) токенов, используемых для аутентификации/авторизации пользователей

Multer — это ПО промежуточного слоя для Express.js. Он сохраняет промежуточные файлы либо в памяти, либо на диске и заполняет объект `res.files` для получения файлов.[2]

#### 1.2.4. Выбор сервера

CentOS — это бесплатный дистрибутив Linux с открытым исходным кодом, созданный на основе Red Hat Enterprise Linux (RHEL).[7]

Преимущества CentOS:

- предоставляет надёжное и проверенное программное обеспечение и обновления для стабильности и безопасности системы;
- имеет `yum`, который позволяет легко обновлять и устанавливать ПО.

В качестве серверной ОС CentOS используется для:

- развёртывания веб-серверов;
- развёртывания баз данных;
- развёртывания сетевого оборудования;
- развёртывания различных сервисов.[17]

Для работы любого сайта нужен веб-сервер —сервис, которая принимает запросы пользователей, обрабатывает их и отправляет обратно ответ. Один из самых популярных сегодня веб-серверов — Nginx.

Nginx — это ПО с открытым исходным кодом для создания мощного и лёгкого веб-сервера.

Nginx решает проблему снижения производительности с повышением роста трафика и является одним из самых популярных веб-серверов в России, занимая 2 место.

Nginx используется:

- Самостоятельно принимать, обрабатывать и отдавать клиентам запросы;
- Выступать в качестве прокси-сервера;

- Для обслуживания серверов, на которые поступает много запросов одновременно;

PM2 - это среда выполнения и диспетчер процессов для Node.js приложений. Это позволяет вам поддерживать приложения активными, перезагружать их без перерывов и облегчать обычные задачи DevOps.[16]

### **1.2.5. Выбор архитектуры**

Для разработки сервера выбран архитектурный стиль программирования MVC, так как он имеет следующие ключевые особенности: чёткое разделение логики представления и логики приложения, существенно уменьшается сложность больших приложений. Код получается гораздо структурированным, и, облегчается его поддержка, тестирование и повторное применение в других частях проекта. Так же, выбранный модуль Node.js подразумевает модель MVC как наиболее эффективную для разработки приложений на его основе.[18]

### **1.2.6. Выбор СУБД**

Система управления базами данных (СУБД) – это набор инструментов, которые позволяют удобно управлять базами данных: удалять, добавлять, фильтровать и находить элементы, менять их структуру и создавать резервные копии.

В разрабатываемой автоматизированной системе будут размещаться результаты оценивания студентов. В каждом годе есть несколько групп, в каждой группе несколько студентов. Так же пользователь может видеть результаты как каждого студента, так и всей группы. По данному описанию функционала работы, становится понятно, что между сущностями базы данных существуют явные связи, что говорит о том, что выбираемая СУБД должна быть реляционной.

Среди распространенных реляционных СУБД существуют два «гиганта»: PostgreSQL и MySQL.

В отличие от MySQL, PostgreSQL является объектно-реляционной системой управления базами данных, что позволяет хранить в ней, к примеру, JSON объект



и осуществлять поиск по значениям его ключей. Также PostgreSQL поддерживает индексирование по выражению, возможность реализовать который может оказаться полезным в случае, когда индексировать необходимо не значение ячейки таблицы, а результат выполнения функции над данным значением.[21]

В качестве СУБД, таким образом, было решено использовать PostgreSQL, так как она распространена, информацию по ее использованию легко найти из-за большого количества документаций, что делает её более удобной для построения веб-приложения в рамках данной ВКР.[11]

### **1.3. Анализ требований технического задания**

В техническом задании были указаны следующие требования:

#### **1.3.1. Функциональные требования**

Программный продукт используется для работы ГЭК и должен обеспечивать выполнение следующих функций:

##### **1.3.1.1. Регистрация пользователей**

Должна быть реализована регистрация пользователей с ролями «Председатель» и «Член ГЭК» через Админ-панель доступную для секретаря. Должно хранить в себе год работы пользователя, email, логин, пароль, роль.

##### **1.3.1.2. Авторизация**

Должна быть реализована авторизация секретаря, председателя и членов ГЭК с разными доступами к сайту.

##### **1.3.1.3. Загрузка и скачивание ВКР**

Должна быть реализована загрузка секретарем ПЗ, ТЗ ВКР в формате pdf

Должно быть реализовано скачивание ПЗ, ТЗ секретарем и председателем на вкладке «ВКР»

##### **1.3.1.4. Загрузка и скачивание Приказов**

Должна быть реализована загрузка секретарем Документации в формате pdf  
Должно быть реализовано скачивание Документации секретарем и председателем на вкладке «Приказы»

##### **1.3.1.5. Добавление, редактирование, удаление групп**

Должно быть реализовано создание, редактирование и удаление групп через Админ-панель, доступную для секретаря

#### 1.3.1.6. Добавление/редактирование/ удаление студентов

Должно быть реализовано создание, редактирование и удаление студентов через Админ-панель, доступную для секретаря.

#### 1.3.1.7. Работа с оценочными листами

Должно быть реализовано заполнение, редактирование оценочного листа по каждому студенту

#### 1.3.1.8. Подведение результатов

Должен быть реализован подсчет результатов по каждому студенту каждым членом ГЭК, общий балл всеми членами ГЭК и итоговая оценка за ВКР. Должен быть реализован подсчет результатов всей группы с итоговыми результатами.

### 1.3.2. Требования к интерфейсу

- ПИ должен поддерживать русский язык
- ПИ должен содержать две вкладки в режиме члена ГЭК «Оценочный лист» и «Результаты», в которых должно быть реализовано заполнение/ редактирование оценочного листа и результаты студентов
- ПИ в режиме администратора должен предоставлять доступ к созданию, редактированию, удалению года, группы, студента, пользователя(члена ГЭК или председателя. Должен предоставлять доступ к добавлению, удалению документации или пояснительные записки, техническое задание ВКР.
- ПИ должен учитывать авторизацию в конфигурации за одну из двух ролей: секретарь. председатель и член ГЭК
- ПИ должен выводить результаты в виде таблиц на группу, на каждого студента

### 1.3.3. Требования к реализации

Для разработки программы должны быть использованы следующие средства:

- Среда разработки WebStorm

- Языки программирования: JavaScript
- Платформа для использования JS на стороне сервера: Node.js
- Фреймворк: Express
- Язык разметки: HTML
- Шаблонизатор: Handlebars
- База данных: PostgreSQL
- GitHub
- Хостинг: TimeWeb Cloud
- VDS-сервер
- Серверная ОС: CentOS 9
- Веб-серверы: Nginx и PM2

#### **1.3.4. Требования к надежности**

- При вводе неправильного логина или пароля, программа должна выдавать сообщение о некорректности введенных данных
- Система должна уведомлять пользователя об ошибках со стороны сервера, если такие произойдут.
- При возникновении сбоев в работе программы она должна автоматически перезапускаться и быстро восстанавливать свою работоспособность.

#### **1.3.5. Требования к окружению**

##### **1.3.5.1. Аппаратные требования**

Минимальная конфигурация ПК:

- Процессор с тактовой частотой не менее 1 ГГц
- Видеоадаптер с объемом видеопамяти не менее 128 Мб
- Монитор
- Клавиатура
- Мышь
- Объем оперативной памяти: 256 Мб
- Объем свободного дискового пространства: 70 Мб

#### 1.3.5.2. Программные требования

Минимальные программные требования:

- Операционная система: Windows 7 или более поздние версии
- Один из интернет-браузеров: Google Chrome, Opera, Internet Explorer, Яндекс Браузер, Mozilla FireFox и тд.

#### 1.3.6. Требования к установке

Принимающему поставляется ссылка на клиент-сервисное приложение

Для успешной инсталляции и работы программы у пользователя должен быть доступ в интернет и любая программа для просмотра веб-страниц

#### 1.3.7. Требования к безопасности

Программа должна иметь сложный пароль на хостинге и регулярно должна происходить его смена

Программа должна иметь сложный пароль на сервере и регулярно должна происходить его смена

Программа должна быть проверена на работоспособность вашего сайта с включенной защитой от DDoS

## **2. Раздел 2. Практическая составляющая работы**

### **2.1. Реализация**

#### **2.1.1. Реализация базы данных**

Проектирование БД является важной и комплексной задачей. Определение ее структуры, является одной из первостепенных задач. Структура базы данных – это порядок или принцип организации записей в базе данных и их связей.

В разрабатываемой веб-платформе используется реляционная модель представления данных, что означает, что данные имеют определённые связи между собой и организуются в виде таблиц, состоящих из строк и столбцов. Таблицы хранят информацию об объектах, представленных в базе данных.

Входными данными для работы с нашей системой будет база данных, с сущностями:

- 1) года;
- 2) пользователи;
- 3) группы;
- 4) студенты;
- 5) ВКР;
- 6) приказы;
- 7) протоколы;
- 8) листы;

#### **Характеристика базы данных**

Концептуальная модель базы данных — это абстрактное представление данных. Она концентрируется на сущностях, их атрибутах и отношениях. Основная цель концептуальной модели — чётко понять важные требования.

Компоненты концептуальной модели включают атрибуты, отношения и сущности. Атрибуты определяют свойства объектов, сущности - ключевые

концепции в предметной области, а отношения показывают связи между объектами.

Создание концептуальной модели данных включает определение объектов, уточнение атрибутов и связей

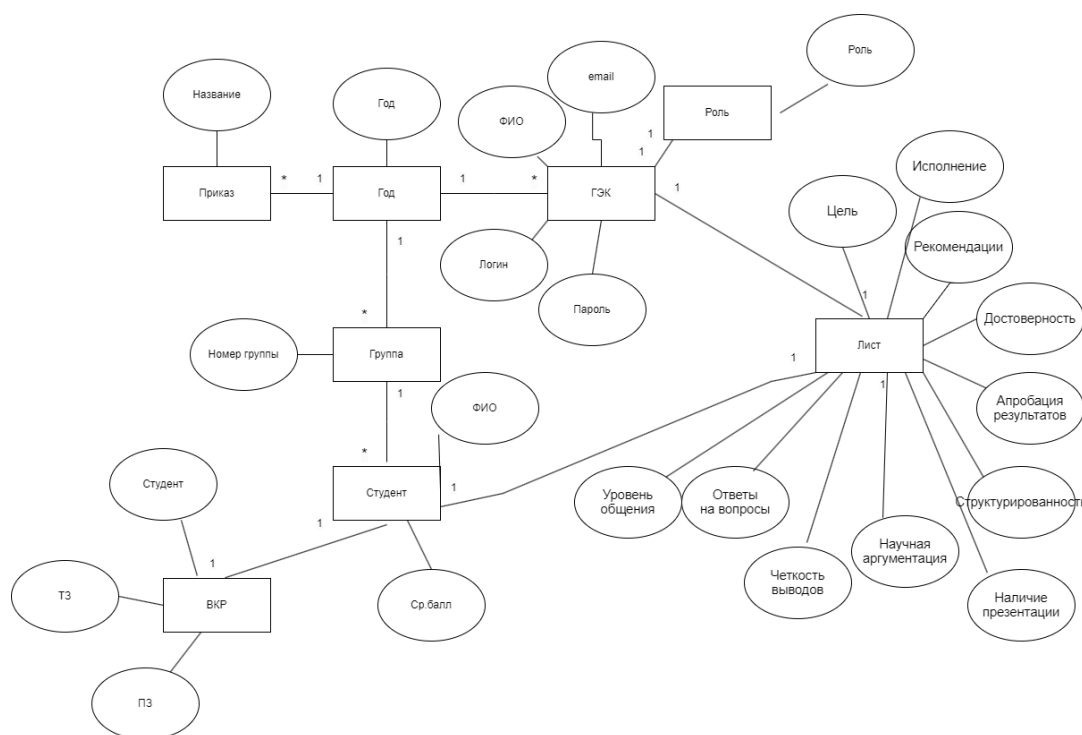


Рисунок 1 -Концептуальная схема базы данных «Работа членов ГЭК»

Согласно теории Т. Кодда были разработаны таблицы для хранения данных, структура которых представлена в таблицах 1-8.

Таблица 1 – Структура таблицы «Года»

Имя поля	Тип данных
Год	Числовой

Таблица 2 – Структура таблицы «Роли»

Имя поля	Тип данных
Роль	Текстовый

Таблица 3 – Структура таблицы «Пользователи»

Имя поля	Тип данных
ФИО	Текстовый
Email	Текстовый
Логин	Текстовый
Пароль	Текстовый
Роль	Текстовый
Год	Числовой

Таблица 4 – Структура таблицы «Группы»

Имя поля	Тип данных
Группа	Текстовый
Год	Числовой

Таблица 5 – Структура таблицы «Приказы»

Имя поля	Тип данных
Название документа	Текстовый
Год	Числовой

Таблица 6 – Структура таблицы «Студенты»

Имя поля	Тип данных
ФИО	Текстовый
Группа	Текстовый
Средний балл	Числовой

Таблица 7 – Структура таблицы «ВКР»

Имя поля	Тип данных
ФИО	Текстовый
Название ПЗ	Текстовый
Название ТЗ	Текстовый

Таблица 8 – Структура таблицы «Оценочный лист»

Имя поля	Тип данных
Цель	Текстовый
Исполнение	Текстовый
Рекомендации	Числовой
Достоверность	Числовой
Апробация результатов	Числовой
Структурированность	Числовой
Наличие презентации	Числовой
Научная аргументация	Числовой
Четкость выводов	Числовой
Ответы на вопросы	Числовой
Уровень общения	Числовой
Студент	Текстовый
Пользователь	Текстовый

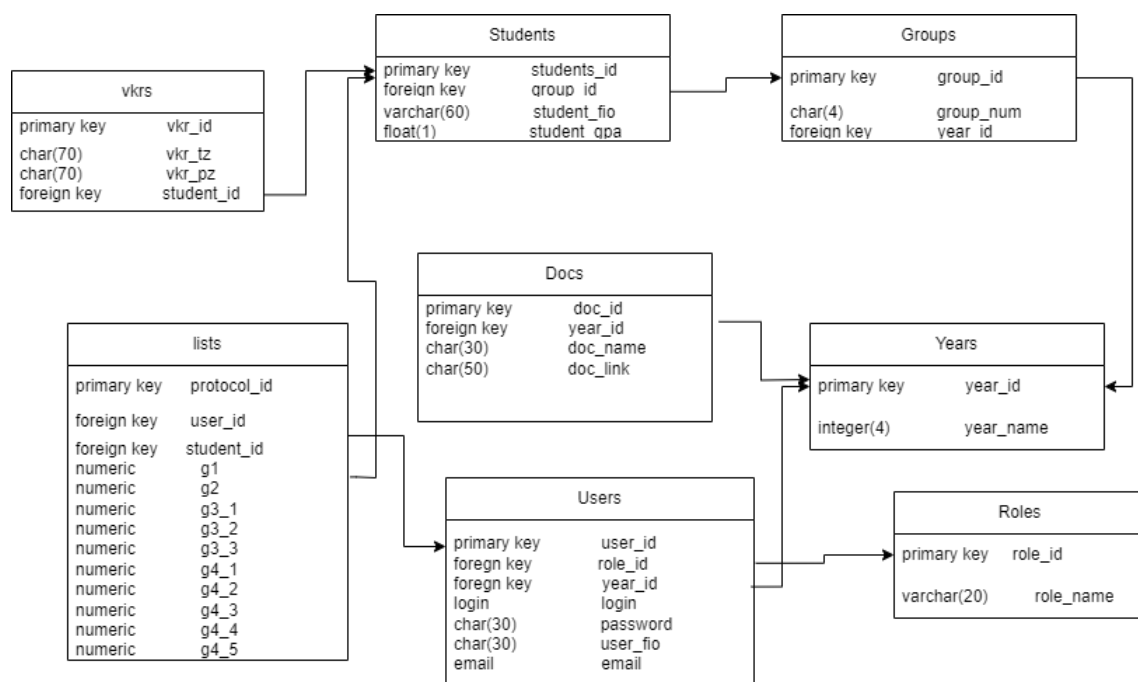


Рисунок 2- Реляционная модель БД



В процессе работы с информационной системой должна предоставляться различная информация:

- список групп;
- список студентов;
- список годов;
- список пользователей;
- список приказов;
- список ВКР;
- результаты группы;
- Результаты оценивания конкретного студента.

По требованию члена ГЭК должна предоставляться следующая информация:

- Заполненный оценочный лист на каждого студента
- Результаты группы;
- Результаты оценивания конкретного студента членами ГЭК.

### **2.1.2. Физическое моделирование АИС**

#### **Выбор архитектуры АИС**

Для реализации системы был выбран архитектурный стиль программирования Model-View-Controller(MVC) – это шаблон проектирования веб-приложений, включающий в себя несколько мелких шаблонов. MVC разделяет прикладную логику, данные и пользовательский интерфейс на три компоненты: контроллер, представление, модель.[9]

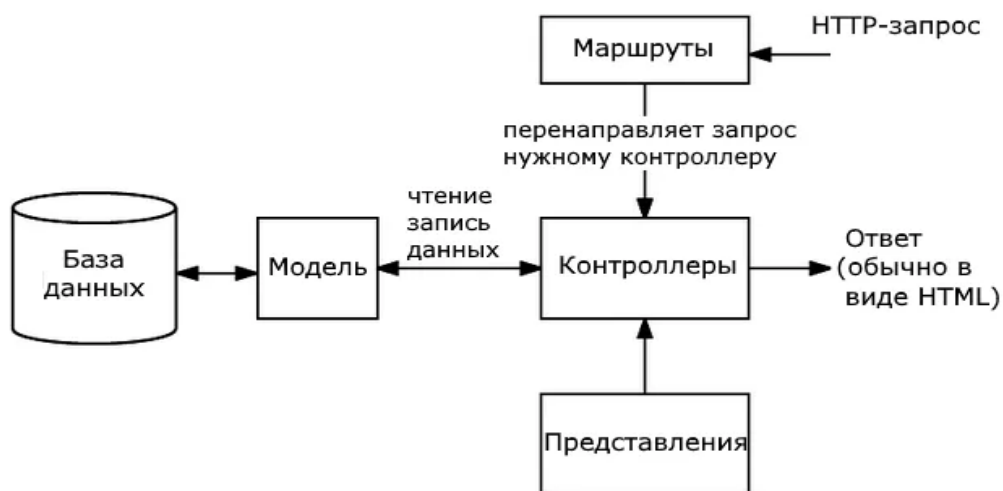


Рисунок 3 – Модель MVC

### Функциональная схема проекта

В составе Государственной экзаменационной комиссии (ГЭК) три роли обеспечивают эффективное управление процессом проведения и гарантируют соблюдение всех необходимых процедур.[20]

Член ГЭК с ограниченными полномочиями. Председатель ГЭК обладает более обширными полномочиями и ответственностью. Секретарь ГЭК имеет права администратора и отвечает за организацию работы комиссии, подготовку и оформление документации. Группы ролей пользователей вы можете увидеть на рисунке 4

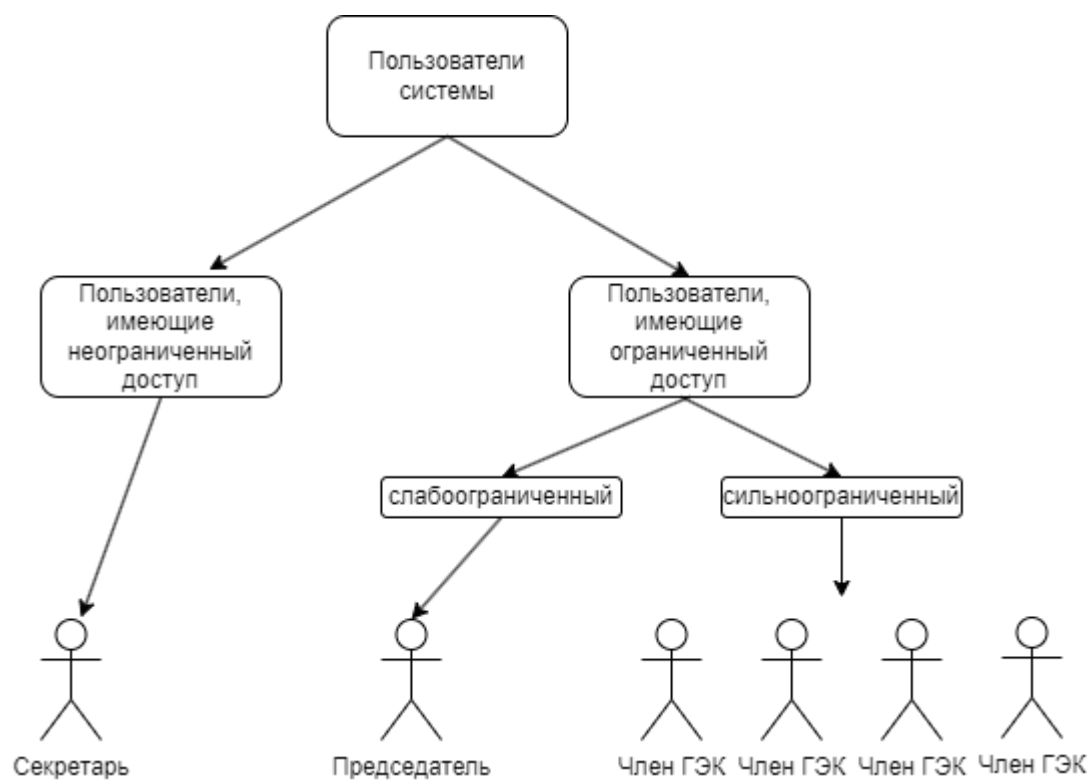


Рисунок 4– Группы пользователей системы

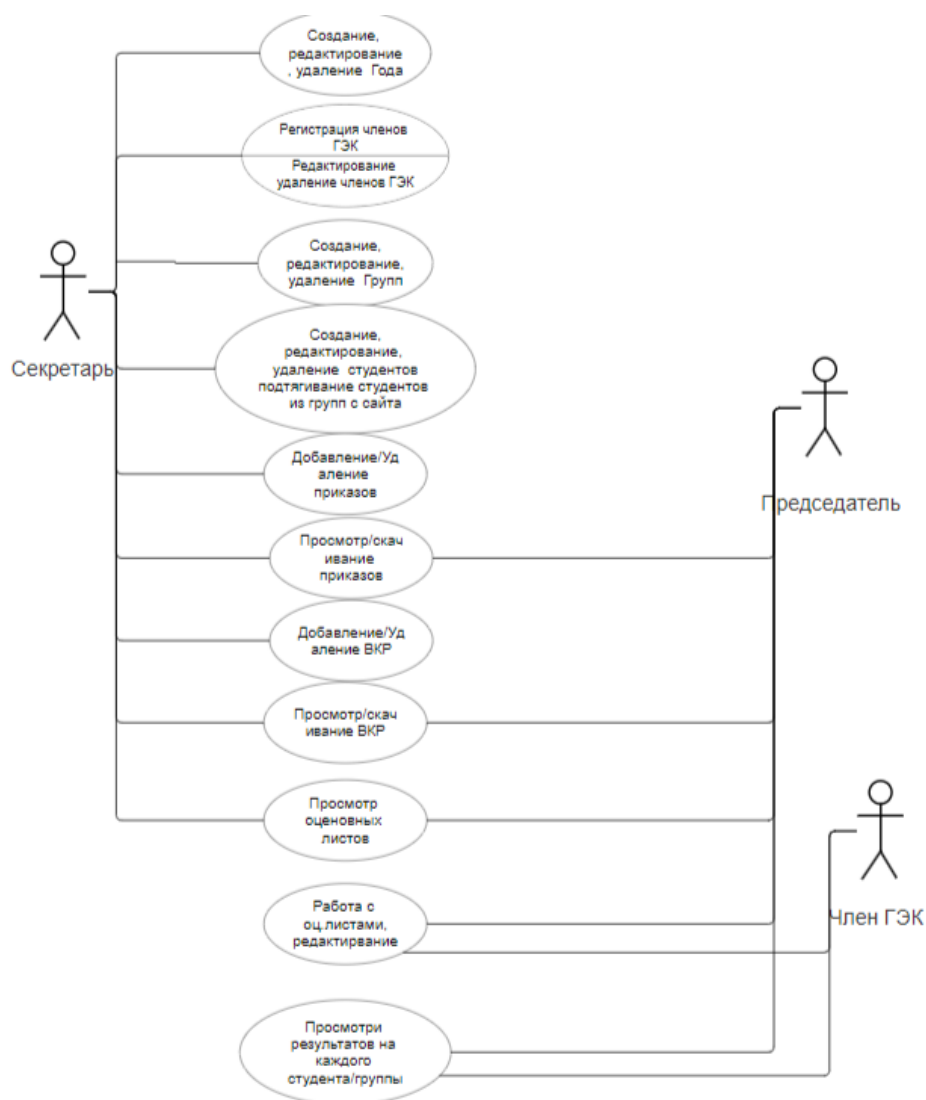


Рисунок 5 – USECASE-диаграмма вариантов использования

Рассмотрим более подробно по каждой из роли ПП.

На рисунке 6 представлена диаграмма возможных состояний программы, выполняющейся под учетной записью члена ГЭК.



Рисунок 6 – Диаграмма возможных состояний программы «Член ГЭК»

При входе в систему член ГЭК совершает только одно действие – авторизовывается в системе, введя логин и пароль, выданный ранее.

После авторизации в систему член ГЭК может выполнить такие функции как:

- выбор группы, студента и заполнения оценочного листа защиты ВКР;
- просмотр результатов оценивания студентов (до заполнения оценочных листов всей ГЭК результатов там не будет).

Диаграмма возможных состояний программы, выполняющейся под учетной записью председателя, приведена на рисунке 7

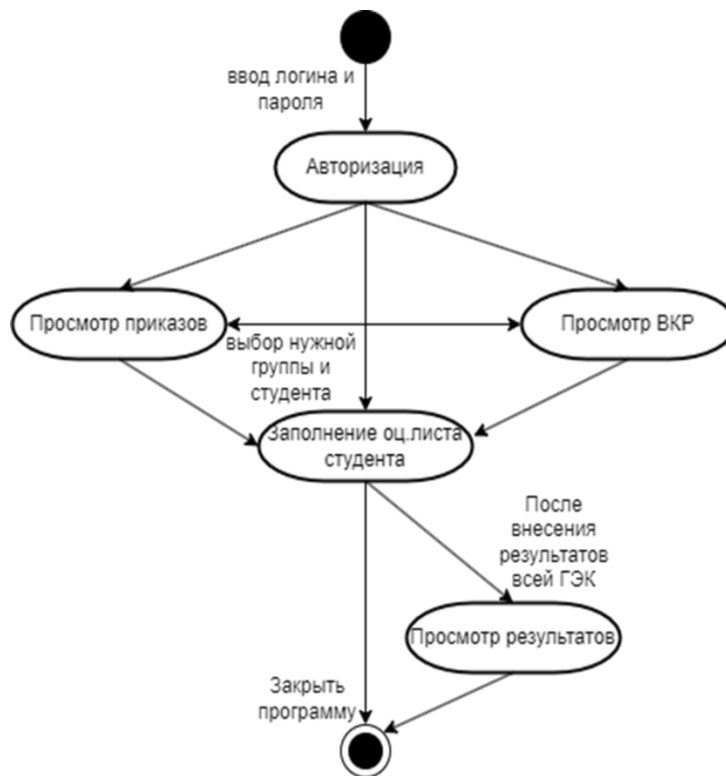


Рисунок 7 – Диаграмма возможных состояний программы  
«Председатель»

При входе в систему Председатель тоже совершает только одно действие— авторизовывается в системе.

После авторизации в систему председатель может выполнить такие функции как:

- просмотр приказов и их скачивание;
- выбор группы, студента и просмотр Пояснительной записки и Технического задания определенного студента;
- выбор группы, студента и заполнения оценочного листа защиты ВКР;
- просмотр результатов оценивания студентов (до заполнения оценочных листов всей ГЭК результатов там не будет).

Диаграмма возможных состояний программы, выполняющейся под учетной записью секретаря на рисунке 8

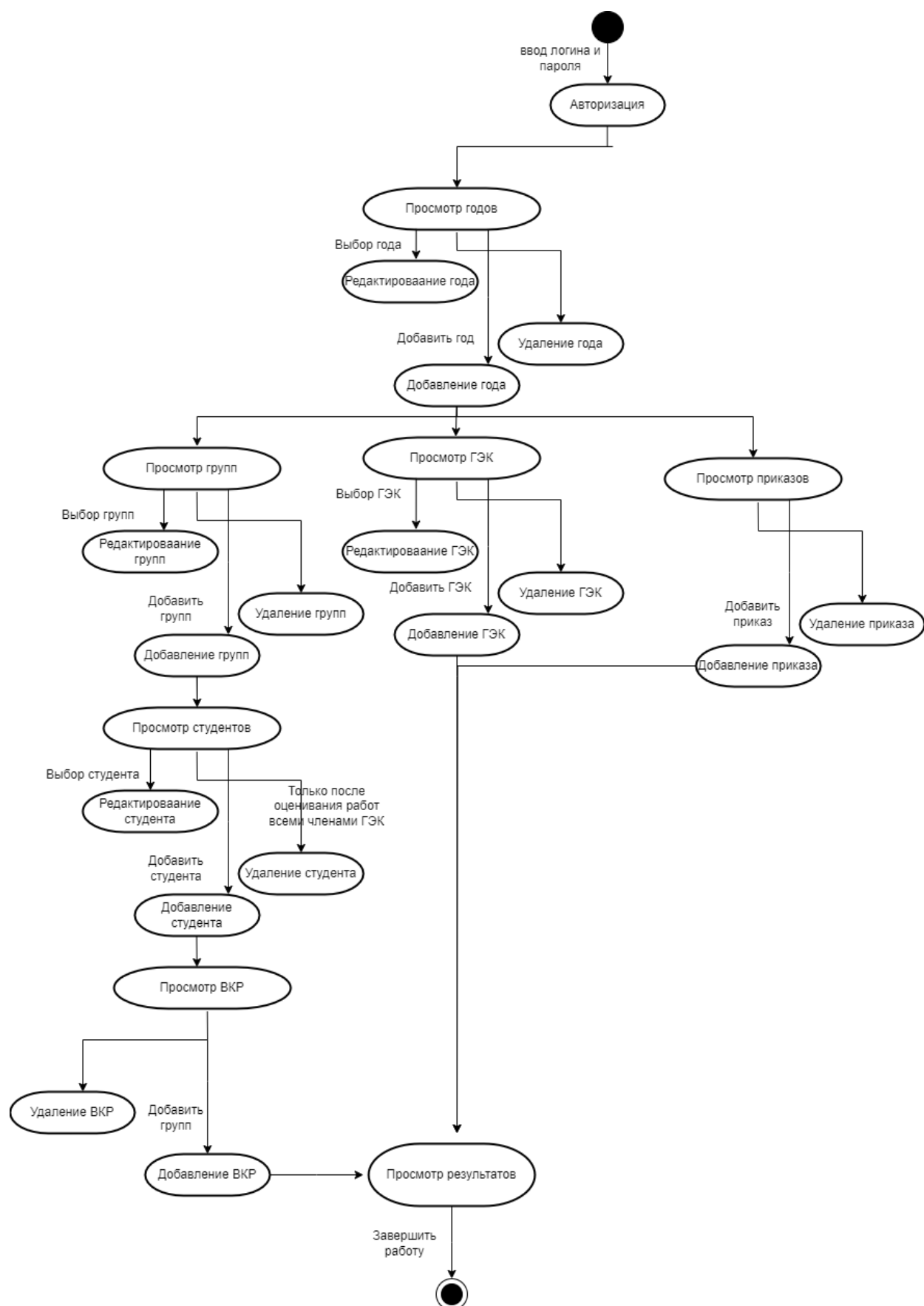


Рисунок 8 – Диаграмма возможных состояний программы «Секретарь»

После авторизации в автоматизированную систему администратор может выполнить такие функции как:

- Просмотр списка года, пользователей, групп, студентов, приказов, ВКР;
- Редактирование годов, пользователей, групп, студентов;
- Добавление и удаление годов, пользователей, групп, студентов, приказов, ВКР;

Просмотр результатов оценивания членами ГЭК;

### Структурная схема проекта

Структурная схема проекта изображена на рисунке 9

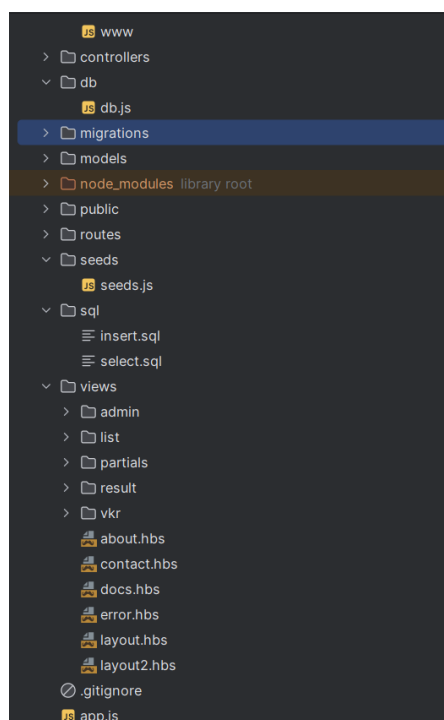


Рисунок 9 – Структурная схема проекта

Описание классов и программных модулей представлено на рисунке 10



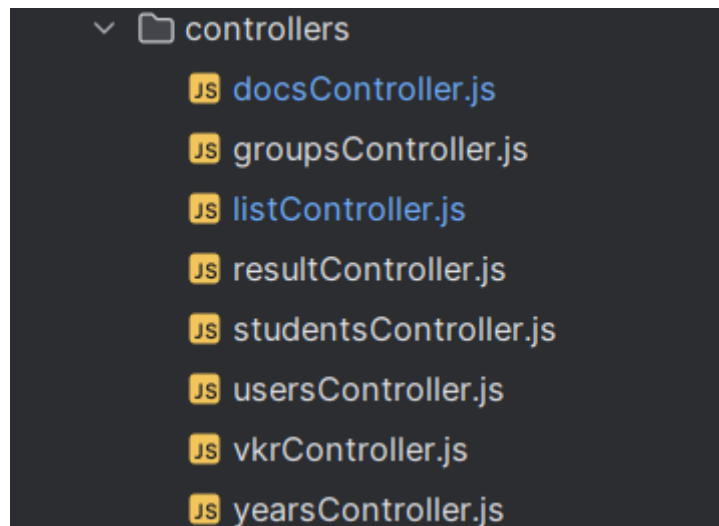


Рисунок 10 – Контроллеры – серверная часть проекта

### Описание программных модулей

Контроллеры, созданные при разработке АИС :

UsersController – класс, для различной работы с Пользователями.

Методы, реализуемые данным классом:

- getAll – отображение всех зарегистрированных пользователей в Админ-панели.
- newUser – для заполнения формы регистрации пользователя
- create – завершение регистрации пользователя.
- getById – вывод одного пользователя по уникальному идентификатору.
- update – редактирование данных пользователя.
- delete – удаления пользователя.
- logout – выхода из программы под этим пользователем.
- Auth – авторизация пользователя.

YearsController – класс, для различной работы с годами.

Методы, реализуемые данным классом:

- yearsAll – отображение всех годов в Главное.
- getAll – отображение всех годов в Админ-панели.
- newYear – для заполнения формы создания года.

- create – завершение добавления года.
- getById – вывод одного года по уникальному идентификатору.
- update – редактирование данных года.
- delete – удаление года;

GroupsController – класс, для работы с группами.

Методы, реализуемые данным классом:

- getAll – отображение всех групп в Админ-панели.
- newGroup – для заполнения формы создания группы.
- create – завершение добавления группы.
- getById – вывод одной группы по уникальному идентификатору.
- update – редактирование данных групп.
- delete – удаление групп;

StudentsController – класс, для работы со студентами.

Методы, реализуемые данным классом:

- getAll – отображение всех студентов в Админ-панели.
- newStudent – для заполнения формы создания студентов.
- create – завершение добавления студентов.
- getById – вывод одного студента по уникальному идентификатору.
- update – редактирование данных студента.
- delete – удаление студентов;

VkrsController – класс, для работы с группами.

Методы, реализуемые данным классом:

- getAll – отображение всех ВКР в Админ-панели.
- vkrsAll – отображение ВКР в «ВКР» в определенной группе.
- studentsAll – отображение списка студентов в определенной группе.
- groupsAll – отображение списка групп.
- newVkr – для заполнения формы добавления ВКР.
- uploadFile – добавление ВКР в локальное хранилище.

- delete – метод удаления ВКР;
- docsController – класс, для работы с группами.

Методы, реализуемые данным классом:

- getAll – отображение всех документов в Админ-панели.
- docsAll – отображение документов в «Приказы».
- studentsAll – отображение списка студентов в определенной группе.
- groupsAll – отображение списка групп.
- newDoc – метод для заполнения формы добавления приказа.
- uploadFile – добавление документов в локальное хранилище.
- delete – удаление документов;

ListsController – класс, для работы с оценочными листами.

Методы, реализуемые данным классом:

- studentsAll – отображение списка студентов в определенной группе.
- groupsAll – отображение списка групп.
- newList – для заполнения формы создания оценочного листа.
- create – метод завершения создания оценочного листа.
- Update – редактирование данных студента.

ResultController – класс, для работы с оценочными листами.

Методы, реализуемые данным классом:

- studentsAll – отображение списка студентов в определенной группе.
- groupsAll – отображение списка групп.
- studentBy – отображение результата студента по уникальному идентификатору;

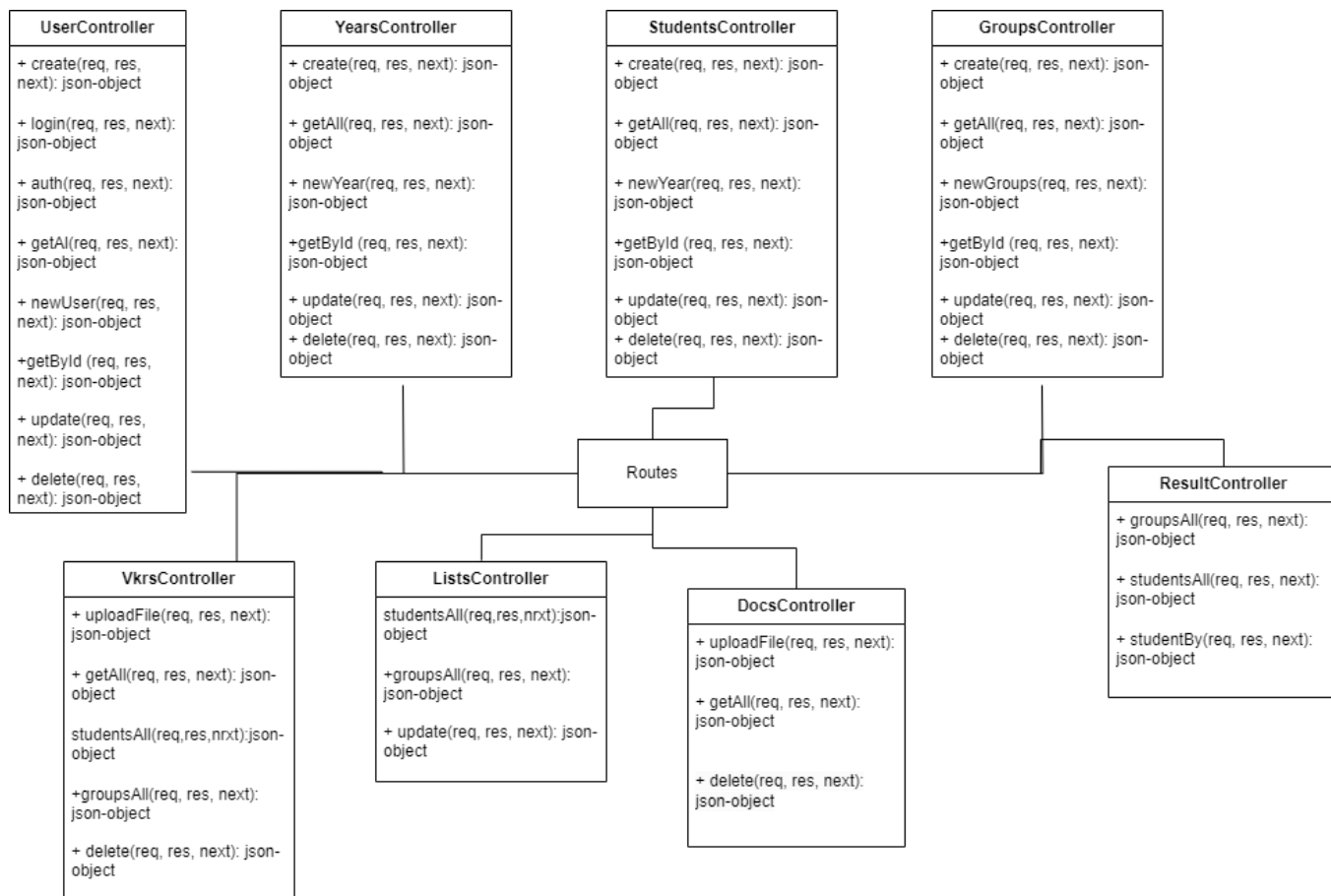


Рисунок 11– Диаграмма классов серверной части.

### 2.1.3. Реализация интерфейса

Админ Панель Секретаря состоит из 7 вкладок в меню. На вкладках «Года», «Группы», «Студенты» происходит добавление, редактирование, удаление данных. В них так же реализовано каскадное удаление данных, удаляя группу, вы удаляете всех студентов, находящихся в ней и все их оценочные листы, поэтому была добавлена форма для подтверждения удаления.

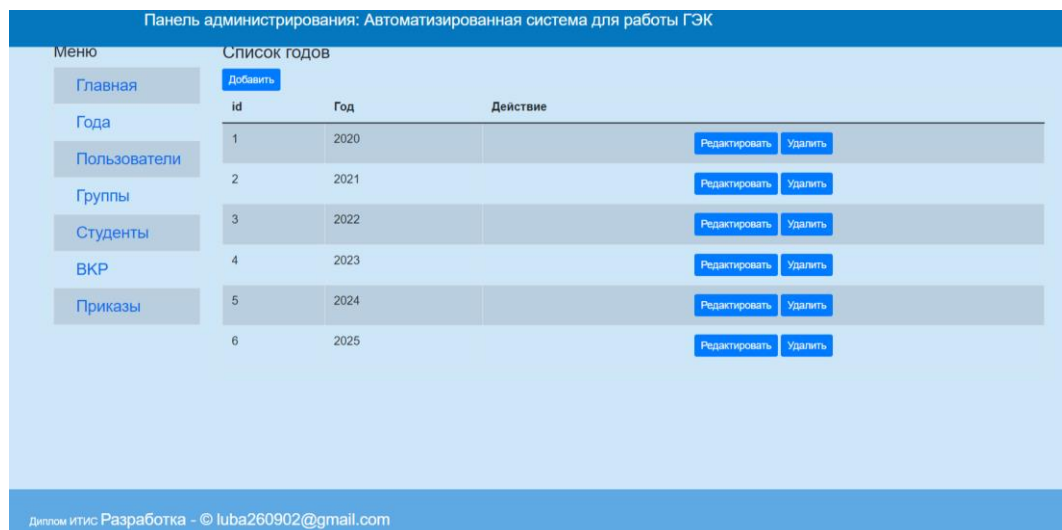


Рисунок 12 – Добавление года в Админ-Панели

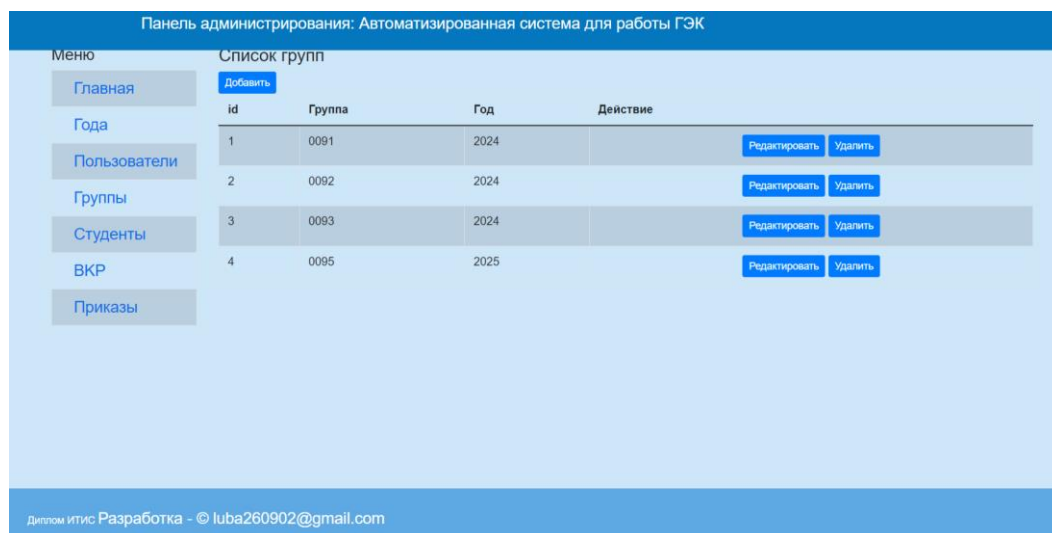


Рисунок 13– Добавление группы в Админ-Панели

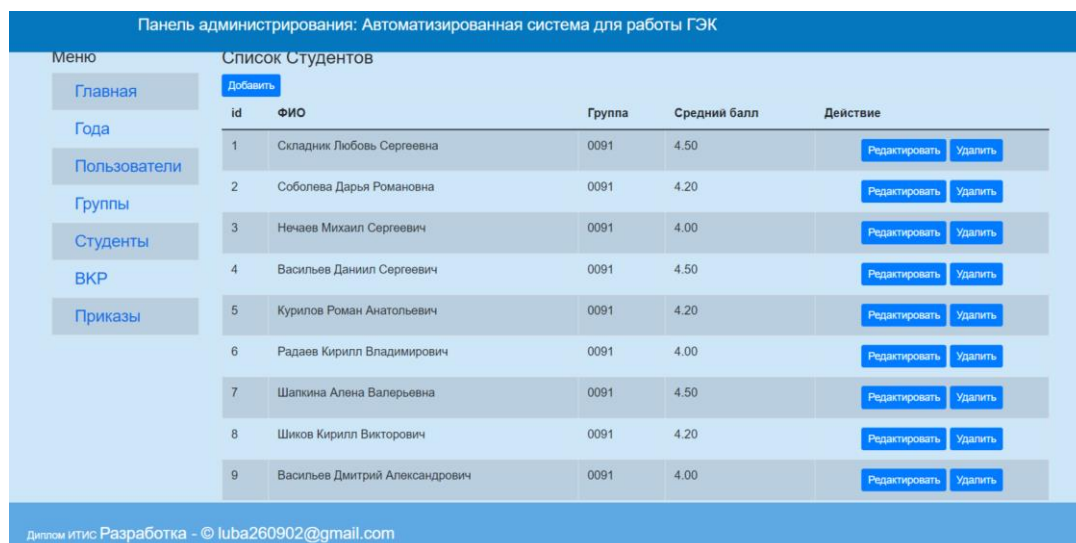
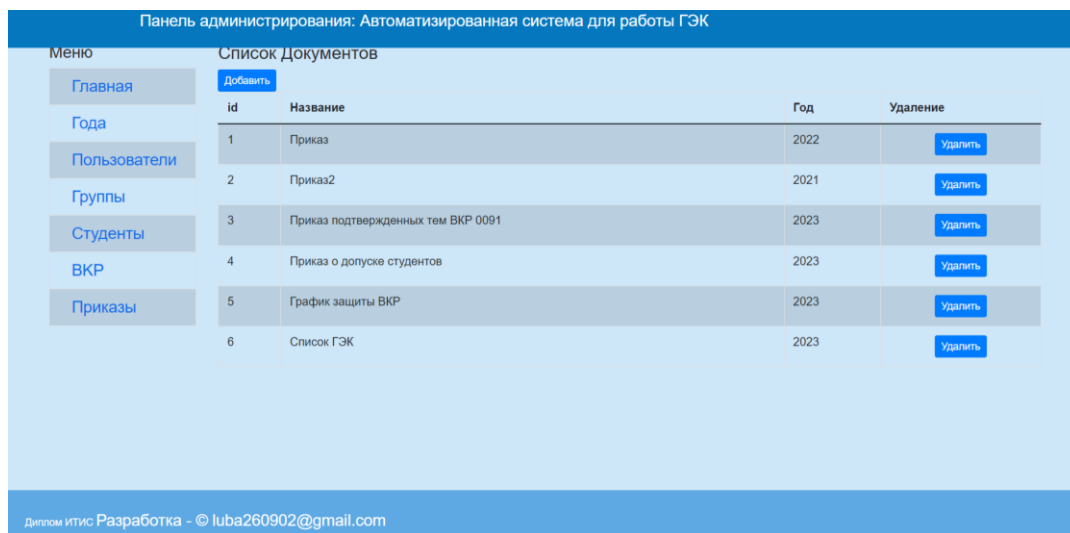


Рисунок 14– Добавление студента в Админ-Панели

В «Пользователи» происходит регистрация пользователей с двумя ролями

В «ВКР», «Приказы» происходит добавление и удаление документов через upload file.



Панель администрирования: Автоматизированная система для работы ГЭК				
Меню Главная Года Пользователи Группы Студенты ВКР Приказы	Список Документов			
	Добавить			
	id	Название	Год	Удаление
	1	Приказ	2022	Удалить
	2	Приказ2	2021	Удалить
	3	Приказ подтвержденных тем ВКР 0091	2023	Удалить
	4	Приказ о допуске студентов	2023	Удалить
	5	График защиты ВКР	2023	Удалить
	6	Список ГЭК	2023	Удалить
	Диплом ИТИС Разработка - © luba260902@gmail.com			

Рисунок 15– Добавление приказов в Админ-Панели

В вкладке «Главная» находятся все года под ссылкой для перехода в нужный год с добавленными данными документациями, группами, студентами и их ВКР.

Меню после авторизации председателя выглядит как показано на рисунке.

Он может скачать нужную документацию для проведения защиты ВКР в «Приказы».

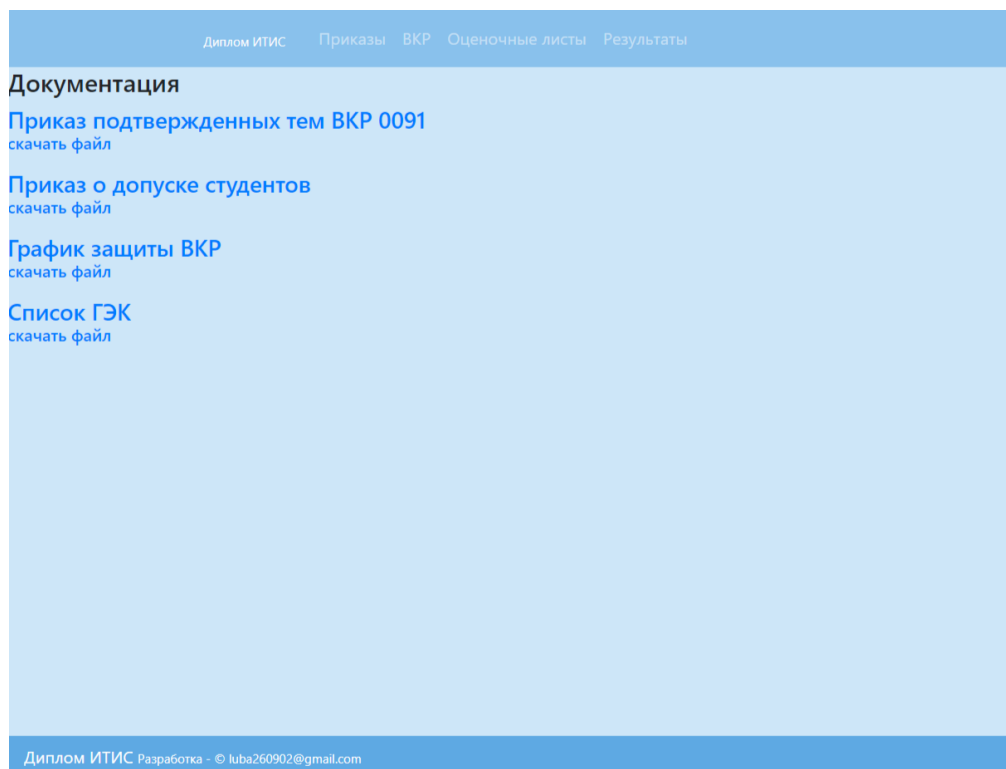


Рисунок 16 – скачивание документации, связанной с этим годом

Так же ему дан доступ для скачивания пояснительных записок и технических задания студентов в вкладке «ВКР».

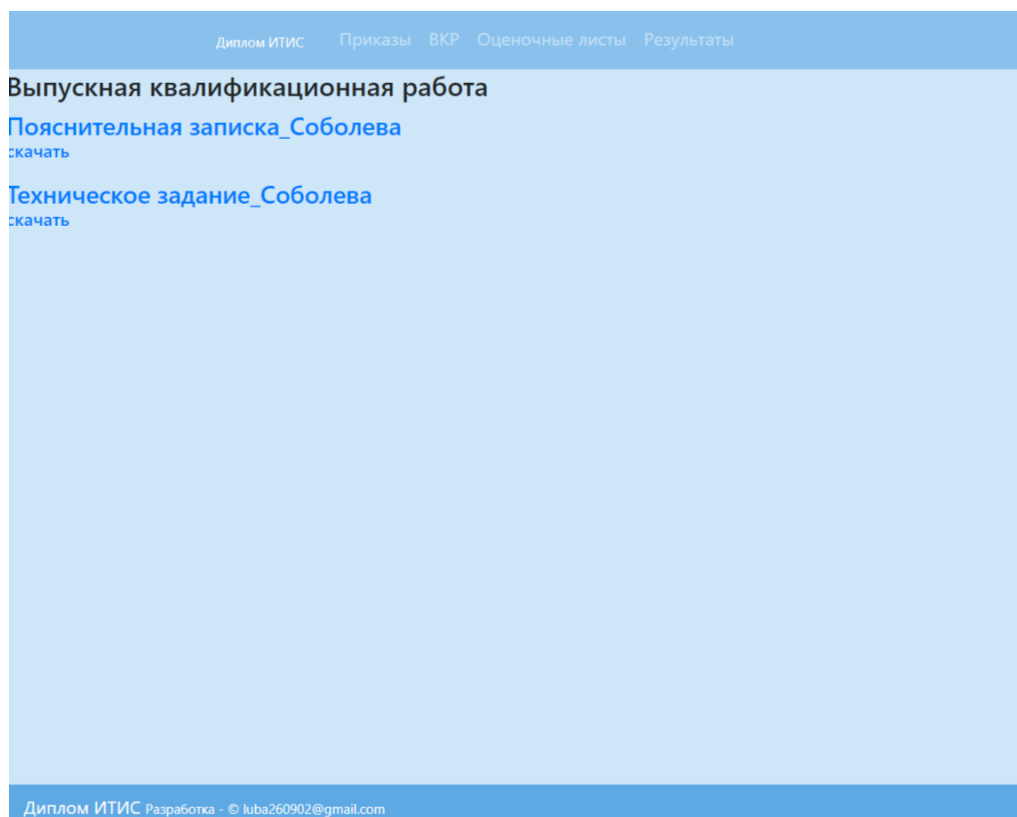


Рисунок 17– Скачивание ПЗ и ТЗ каждого студента

Заполнять/редактировать оценочные листы на каждого студента и просматривать результаты.

Диплом ИТИС Приказы ВКР Оценочные листы Результаты			
<b>Оценочный лист</b>			
Добавить Редактировать			
Наименование	Показатели освоения компетенции согласно ФГОС и ОПОП	Максимальный балл	Оцениваемый балл
Постановка цели и задач исследования	Актуальность и целевая направленность работы	5	5.0
	2. Обоснованность и полнота сформулированных задач в соответствии с утвержденной темой		
	3. Применение эффективных методик использования программных средств решения поставленных задач		
Исполнение	1. Полнота и структурированность работы; умение научно аргументировать свою позицию, использовать основные законы естественнонаучных дисциплин в профессиональной деятельности	5	4.5
	2. Умение логически верно и грамотно строить письменную речь		
	3. Использование современных инструментальных средств и технологий программирования		
	4. соответствие оформления работы действующему стандарту организации СТО 1.701-2010, требованиям проверки на предмет заимствования		
Результаты	1. Наличие практических рекомендаций по решению поставленных в работе задач, наличие методик использования программного продукта для программиста и пользователя	5	4.8
	2. Достоверность и обоснованность выводов по проведенному исследованию, их соответствие заявленной цели	5	2.7

Рисунок 18– Заполненный оценочный лист студента

Меню после авторизации члена ГЭК выглядит как показано на рисунке.

Есть только Вкладка «Оценочные листы» и «Результаты»

«Оценочный лист» определенного студента на рисунке

В вкладке «Результаты» можно просматривать итоговые результаты всей группы и каждого студента более подробно по отдельности (рисунки 19 и 20).

Диплом ИТИС Приказы ВКР Оценочные листы Результаты		
<b>Результаты</b>		
id	ФИО	Итоговая оценка
1	Складник Любовь Сергеевна	4.5
2	Соболева Дарья Романовна	4.2
3	Нечаев Михаил Сергеевич	4.2
4	Васильев Даниил Сергеевич	4.3
5	Курилов Роман Анатольевич	4.3
6	Радаев Кирилл Владимирович	4.2
7	Шапкина Алена Валерьевна	4.4

Диплом ИТИС. Разработка - © kuba260902@gmail.com

Рисунок 19– результаты группы



Диплом ИТИС Приказы ВКР Оценочные листы Результаты	
Результаты	
Нечаев Михаил Сергеевич	
ФИО проверяющего	Средний балл
Цымбалюк Лариса Николаевна	4.5
Максимов Виктор Андреевич	4.3
Макаров Николай Антонович	4.3
Итоговая оценка	4.4

Диплом ИТИС Разработка - © kuba260902@gmail.com

Рисунок 20– Результат определенного студента

Диаграмма «как есть» — это графическое представление текущего состояния системы или процесса. Она показывает существующие связи между элементами системы, потоки данных и ресурсов, а также выявляет проблемные области и узкие места. Диаграмма «как есть» служит основой для дальнейшего анализа и разработки улучшений.



Рисунок 21 – Диаграмма «Как есть»

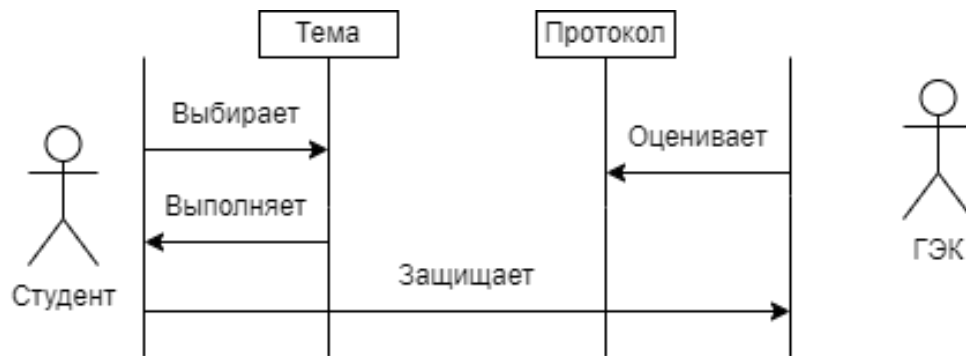


Рисунок 22 – UML-диаграмма процесса оценки ВКР студентов

## 2.2. Тестирование

QA- это обеспечение качества. Огромную роль в обеспечении качества(QA) проекта играет тестирование.[13]

В этом проекте мы будем использовать различные методы тестирования:

- Страничное тестирование. Страничное тестирование, как понятно из названия, тестирует функциональность и визуализацию страницы на стороне клиента. Может включать как интеграционное, так и модульное тестирование. Для этих целей лучше использовать фреймворк Mocha.
- Линтинг. Он относится к обнаружению потенциальных или будущих ошибок. Общая цель линтинга: нахождение частей кода, потенциально написанных, не соблюдая правила написания кода, выделяя участки с ошибками. Что очень облегчает работу с проектом, и отслеживание создающихся ошибок, до их выявления, для их устранения.
- Проверка ссылок. Проверка ссылок – цель которых удостовериться в отсутствии нерабочих ссылок на сайте. Чаще всего их проверка, делается в крупнейших проектах. Но лучше даже в легких проектах, где легко это проверить в ручную, включать проверку ссылок. Для проверки используется LinkChecker

В мой проект будут интегрированы 4 вида тестов: страничные, логические, линтинг, проверка ссылок

### 2.2.1. Набор тестов

1) Страничное тестирование бывает двух типов: глобальное и страницезависимое.

Глобальные тесты должны иметь общие условия для всех страниц сайта, то есть проверка общих параметров:

2) Линтинг. Как и у естественных языков, у языков программирования тоже есть свои правила и синтаксис. Линтер следит за тем, чтобы они соблюдались.

Важный момент: линтер нужно настроить. Настроенный линтер позволяет унифицировать код, который пишут разные люди при работе над конкретным проектом. Часто для этого используют один конкретный линтер для одного языка программирования.

Линтер не находит багов — если программист неправильно написал код, он не укажет на проблему. Но укажет место потенциальных ошибок из-за нарушения синтаксиса языка.

Есть много разных линтеров для каждого языка программирования. Они отличаются по глубине настройки и возможностям анализа кода — как глубоко они могут его анализировать. Например, для JavaScript есть JSLint, JSHint и ESLint — это всё линтеры, которые делают плюс-минус одно и то же и различаются в деталях.

Вот основные отличия между инструментами ESLint, JSLint и JSHint:

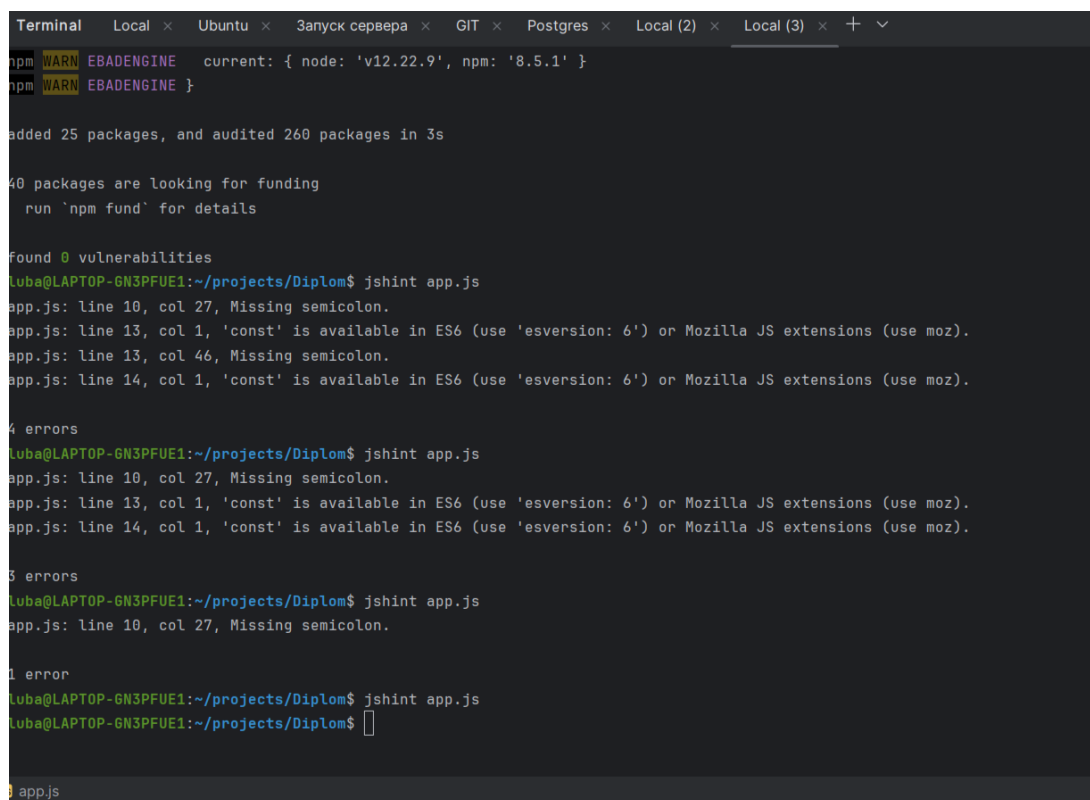
ESLint — это инструмент, который легко настраивается и поддерживает последние функции ECMAScript. Он также позволяет использовать собственные пользовательские правила.

JSLint — это инструмент, который является более строгим и не допускает значительной настройки. Он устанавливает строгий стиль кодирования, что

может быть полезно для поддержания согласованности в больших кодовых базах.

JSHint — это более гибкая альтернатива JSLint. Он допускает некоторую настройку и поддерживает больше функций ECMAScript, чем JSLint.

В моем проекте используется JSHint из-за возможности настроить его под свои запросы.



```
Terminal Local x Ubuntu x Заняск сервера x GIT x Postgres x Local (2) x Local (3) x + v
npm WARN EBADENGINE current: { node: 'v12.22.9', npm: '8.5.1' }
npm WARN EBADENGINE }

added 25 packages, and audited 260 packages in 3s

40 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
Luba@LAPTOP-GN3PFUE1:~/projects/Diplom$ jshint app.js
app.js: line 10, col 27, Missing semicolon.
app.js: line 13, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
app.js: line 13, col 46, Missing semicolon.
app.js: line 14, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

4 errors
Luba@LAPTOP-GN3PFUE1:~/projects/Diplom$ jshint app.js
app.js: line 10, col 27, Missing semicolon.
app.js: line 13, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
app.js: line 14, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).

3 errors
Luba@LAPTOP-GN3PFUE1:~/projects/Diplom$ jshint app.js
app.js: line 10, col 27, Missing semicolon.

1 error
Luba@LAPTOP-GN3PFUE1:~/projects/Diplom$ jshint app.js
Luba@LAPTOP-GN3PFUE1:~/projects/Diplom$ 
app.js
```

Рисунок 23– Проверка проекта Линтером

#### 4) Проверка ссылок

LinkChecker — программа под Linux для поиска неработающих ссылок на сайте. Программа предназначена, прежде всего, для веб-мастеров и администраторов сайтов.

Достаточно указать адрес сайта и нажать кнопку Start. LinkChecker начнет сканировать сайт — просматривать все внешние и внутренние ссылки и

проверять их доступность. В процессе работы формируется список с результатами. По каждой ссылке можно получить информацию.

linkchecker http://localhost:3000

```
luba@LAPTOP-GN3PFUE1:~/projects/Diplom$ linkchecker http://localhost:3000
INFO linkcheck.cmdline 2024-06-05 11:52:59,825 MainThread Checking intern URLs only; use --check-extern to check ext
LinkChecker 10.0.1
Copyright (C) 2000-2016 Bastian Kleineidam, 2010-2021 LinkChecker Authors
LinkChecker comes with ABSOLUTELY NO WARRANTY!
This is free software, and you are welcome to redistribute it under
certain conditions. Look at the file 'LICENSE' within this distribution.
Get the newest version at https://linkchecker.github.io/linkchecker/
Write comments and bugs to https://github.com/linkchecker/linkchecker/issues

Start checking at 2024-06-05 11:52:59+003

URL      'http://localhost:3000'
Real URL  http://localhost:3000
Check time 0.013 seconds
Result    Error: ConnectionError: HTTPConnectionPool(host='localhost', port=3000): Max retries exceeded with url: /
on.HTTPConnection object at 0x7f5373e29b70>: Failed to establish a new connection: [Errno 111...]

Statistics:
Downloaded: 0B.
Content types: 0 image, 0 text, 0 video, 0 audio, 0 application, 0 mail and 1 other.
URL lengths: min=21, max=21, avg=21.
```

Рисунок 24 – Демонстрация проверки ссылок

## **Заключение**

В рамках выполнения ВКР была создана автоматизированная система работы ГЭК – оценивание комиссией выпускных квалификационных работ студентов.

Затронутая тема является актуальной, т.к. на сегодняшний день система сдачи ВКР очень тяжелый труд для секретарей ГЭК, нужно создать удобные, несложные условия для оценивания работ студентов выпускных курсов, автоматизировать эту работу до автоматического подсчета баллов и итоговой оценки студента.

Автоматизированные системы для работы ГЭК существенно упрощают защиту ВКР студентов – они способны быстро подвести итоговую оценку за ВКР со условиями оценивания.

В основе системы лежит база данных, разработанная при помощи СУБД PostgreSQL.

В качестве среды разработки автоматизированной системы был выбран WebStorm и язык программирования JavaScript с использованием платформы Node.js

Подводя итог поставленной задачи к ВКР, можно сделать вывод, что разработанная система полностью удовлетворяет поставленным требованиям и может быть использована на нашей кафедре для упрощенного оценивания выпускников, а так же и на других кафедрах, внося только некие правки в оценочные листы.

## Список источников

1. David Herron Node.js Web Development-114 с
2. File uploading in Node [Электронный ресурс] <https://www.geeksforgeeks.org/file-uploading-in-node-js/> (Дата обращения 15.04.2024)
3. Git [Электронный ресурс] <https://git-scm.com/book/ru/v2> (Дата обращения 02.03.2024)
4. Handlebars [Электронный ресурс] <https://handlebarsjs.com/guide/> (Дата обращения 10.03.2024)
5. How to upload files in Node.js and Express [Электронный ресурс] <https://attacomsian.com/blog/uploading-files-nodejs-express> (Дата обращения 28.03.2024)
6. Knex.js Tutorial | A Complete Guide [Электронный ресурс] <https://stackfame.com/knexjs-complete-tutorial> (Дата обращения 27.03.2024)
7. Oliver Pelz, Jonathan Hobson - CentOS 7 Linux Server Cookbook
8. Simon Holmes Getting MEAN with Mongo, Express, Angular, and Node - 56 с
9. Spring MVC — основные принципы / Хабр – Habr [Электронный ресурс]: <https://habr.com/ru/post/336816/> (Дата обращения 10.05.2024)
10. Shelley Powers Learning Node -74 с
11. Using PostgreSQL with Node.js and node-postgres [Электронный ресурс] <https://stackabuse.com/using-postgresql-with-nodejs-and-node-postgres/> (Дата обращения 29.03.2024)
12. Understanding MVC pattern in Nodejs [Электронный ресурс] [https://dev.to/eetukudo\\_/understanding-mvc-pattern-in-nodejs-2bdn](https://dev.to/eetukudo_/understanding-mvc-pattern-in-nodejs-2bdn) (Дата обращения 12.04.2024)
13. Веб-разработка с применением Node.JS и Express/ Полноценное

использование стека JavaScript Итан Браун

14. Демина А.В. Базы данных. Эффективная работа в Access. Использование языка структурированных запросов SQL; Саратовский государственный социально-экономический университет. – Саратов, 2013. – 44 с.
15. Долженко А.И. Управление информационными системами. – Ростов-на-Дону: Изд-во РГУ, 2017. – 191 с
16. Инструкция по настройке PM-2 сервера [Электронный ресурс] <https://pm2.keymetrics.io/docs/usage/quick-start/> (Дата обращения 20.04.2024)
17. Обзор CentOS: версии дистрибутива и его преимущества [Электронный ресурс] <https://selectel.ru/blog/centos/> (Дата обращения 10.04.2024)
18. Паттерн MVC [Электронный ресурс] <https://metanit.com/web/nodejs/7.1.php> (Дата обращения 1.04.2024)
19. Рекомендации по выполнению выпускной квалификационной работы по специальности 230105 – «Программное обеспечение вычислительной техники и автоматизированных систем» Методическое пособие Макаров В.А., Винник Л.И.
20. Роль, цели и задачи государственной аттестационной комиссии (ГАК) [Электронный ресурс] <https://disshelp.ru/blog/rol-tseli-i-zadachi-gak/> (Дата обращения 12.02.2024)
21. Руководство по PostgreSQL [Электронный ресурс] <https://metanit.com/sql/postgresql/> (Дата обращения 29.03.2024)
- 22.. Руководство по Knex [Электронный ресурс] <https://knexjs.org/guide/> (Дата обращения 27.03.2024)



## Приложение А

```
Bin/www.js
#!/usr/bin/env node

var app = require('./app');
var debug = require('debug')('myapp:server');
var http = require('http');

var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

var server = http.createServer(app);

server.listen(port);
server.on('error', onError);
server.on('listening', onListening);

function normalizePort(val) {
  var port = parseInt(val, 10);
  if (isNaN(port)) {
    // named pipe
    return val;
  }

  if (port >= 0) {
    // port number
    return port;
  }
  return false;
}

function onError(error) {
  if (error.syscall !== 'listen') {
    throw error;
  }

  var bind = typeof port === 'string'
    ? 'Pipe ' + port
    : 'Port ' + port;

  // handle specific listen errors with friendly messages
  switch (error.code) {
    case 'EACCES':
      console.error(bind + ' requires elevated privileges');
      process.exit(1);
      break;
    case 'EADDRINUSE':
      console.error(bind + ' is already in use');
      process.exit(1);
      break;
    default:
      throw error;
  }
}

function onListening() {
  var addr = server.address();
  var bind = typeof addr === 'string'
    ? 'pipe ' + addr
```

```

: 'port ' + addr.port;
debug('Listening on ' + bind);
}

db/db.js

const environment = process.env.NODE_ENV || 'development';
const conf = require('../knexfile')[environment];
module.exports = require('knex')(conf);
controllers/docsController.js
const docsModel = require('../models/docsModel');
const yearsModel = require("../models/yearsModel");
const groupsModel = require("../models/groupsModel");
const vkrsModel = require("../models/vkrsModel");
const studentsModel = require("../models/studentsModel");
const multer = require('multer');
const fileUpload= require('../middleware/upload_docs');
// import {Translate} from "translate";

module.exports = docsController = {
  getAll: async (req, res, next) => {
    try {
      const year_id = parseInt(req.params.yearID); // Получаем id_year из параметра маршрута
      const docs = await docsModel.getAll(year_id);
      const years = await yearsModel.getById(year_id)
      res.render('docs', {
        title: 'Документация',
        layout: 'layout2',
        docs,
        years
      });
    } catch (error) {
      next(error);
    }
  },
  docsAll: async (req, res, next) => {
    try {
      const docs = await docsModel.docsAll(); // Передаем id_year в модель для получения
данных
      res.render('admin/docs/docs', {
        title: 'Документация',
        docs
      });
    } catch (error) {
      next(error);
    }
  },
  newDoc: async (req, res, next) => {
    const years= await docsModel.yearsAll()
    try {
      res.render('admin/docs/add', {
        title: 'add new',
        layout: 'layout',
        years
      })
    } catch (error) {
      next(error);
    }
  },

  uploadFile:function(req,res){

```

```

const upload = multer({
  storage: fileUpload.files.storage(),
  allowedFile:fileUpload.files.allowedFile
}).single('file');

upload(req, res, function (err) {
  console.log(req.file.filename);
  console.log(req.body.year_id);

  // const file =  translate(req.file.filename, 'en');

  const file = Buffer.from(req.file.filename, 'latin1').toString('utf8');
  const year_id=req.body.year_id;
  docsModel.create({ file,year_id },(err, uploadeddata)=>
  {
    if(err) return res.send(err);
    console.log(uploadeddata);
  })

  if (err instanceof multer.MulterError) {
    res.send(err);
  } else if (err) {
    res.send(err);
  }else{
    res.render('admin/docs/add');
  }
  res.redirect('/docs');
})
},
delete: async (req, res, next) => {
  try {
    const doc = await docsModel.delete(req.params.id);
    res.render('admin/docs/docs', {
      title: 'Удаление БКР',
      doc
    })
  } catch (error) {
    next(error);
  }
  res.redirect('/docs');
}
};
controllers/groupsController.js
const groupsModel = require('../models/groupsModel');
const studentsModel = require("../models/studentsModel");
const yearsModel = require("../models/yearsModel");

module.exports = groupsController = {

  groupsAll: async (req, res, next) => {
    try {
      const groups = await groupsModel.groupsAll();
      res.render('admin/groups/groups', {
        title: 'Группы',
        groups,
        layout:'layout'
      })
    } catch (error){
      next(error);
    }
  },

```

```

newGroup: async (req, res, next) => {
  const years = await groupsModel.yearsAll()
  try {

    res.render('admin/groups/add', {
      title: 'Добавить группу',
      years
    })
  } catch (error) {
    next(error);
  }
},

create: async (req, res, next) => {
  try {
    const group = await groupsModel.create(req.body);
    res.render('admin/groups/add', {
      title: 'Добавить группу',
      group
    })
  } catch (error) {
    next(error);
  }
  res.redirect('/groups');
},

getById: async (req, res, next) => {
  try {
    const groups = await groupsModel.getById(req.params.id);
    const years= await groupsModel.yearsAll()
    res.render('admin/groups/edit', {
      title: 'Редактирование группы',
      groups,
      years,
    })
  } catch (error) {
    next(error);
  }
},

update: async (req, res, next) => {
  try {
    const group = await groupsModel.update(req.params.id, req.body);
    res.render('admin/groups/edit', {
      title: 'Редактирование группы',
      group
    })
    // res.send(user);
  } catch (error) {
    next(error);
  }
  res.redirect('/groups');
},

delete: async (req, res, next) => {
  try {
    const group = await groupsModel.delete(req.params.id);
    res.render('admin/groups/groups', {
      title: 'Удаление',
      group
    })
  } catch (error) {
    next(error);
  }
}

```

```

    }
    res.redirect('/groups');
  }
};
controllers/listsController.js
const studentsModel = require("../models/studentsModel");
const groupsModel = require("../models/groupsModel");
const listsModel = require("../models/listsModel");
const usersModel = require("../models/usersModel");
const docsModel = require("../models/docsModel");
const vkrsModel = require("../models/vkrsModel");
const yearsModel = require("../models/yearsModel");

module.exports = listController = {
  partial: async (req, res, next) => {
    try {
      const year_id = parseInt(req.params.yearID);
      const docs = await listsModel.groupsAll(year_id);

      // const docs = await docsModel.getAll1(year_id);
      res.render('partial/menu', {
        title: 'Документация',
        docs,
        // docs
      });
    } catch (error) {
      next(error);
    }
  },
  groupsAll: async (req, res, next) => {
    try {
      const year_id = parseInt(req.params.yearID); // Получаем id_year из параметра маршрута
      const groups = await listsModel.groupsAll(year_id);
      const years = await yearsModel.getById(year_id)

      res.render("list/groups", {
        title: 'Список групп',
        layout: 'layout2',
        groups,
        years
      })
    } catch (error) {
      next(error);
    }
  },
  studentsAll: async (req, res, next) => {
    try {
      const year_id = parseInt(req.params.yearID);
      const group_id = parseInt(req.params.groupID);
      const students = await listsModel.studentAll(group_id, year_id);
      const years = await yearsModel.getById(year_id)

      res.render('list/students', {
        title: 'Список студентов',
        layout: 'layout2',
        students,
        years
      })
    }
  }
};

```

```

    } catch (error){
      next(error);
    }
  },
  getAll: async (req, res, next) => {
    try {
      const year_id = parseInt(req.params.yearID);
      const student_id = parseInt(req.params.studentID);
      const lists = await listsModel.getAll(student_id);
      const years = await yearsModel.getById(year_id)
      res.render('list/list', {
        title: 'Оценочный лист',
        layout: 'layout2',
        lists,
        years
      })
    } catch (error){
      next(error);
    }
  },
  newList: async (req, res, next) => {
    try {
      res.render('list/list', {
        title: 'Заполнение оценочного листа',

      })
    } catch (error) {
      next(error);
    }
  },
  create: async (req, res, next) => {
    try {
      const user = await usersModel.create(req.body);
      res.render('admin/users/edit', {
        title: 'Добавление пользователя',
        user
      })
    } catch (error) {
      next(error);
    }
  },
};
controllers/reaultController.js
const studentsModel = require("../models/studentsModel");
const groupsModel = require("../models/groupsModel");
const usersModel = require("../models/usersModel");
const listsModel = require("../models/listsModel");
const yearsModel = require("../models/yearsModel");

module.exports = vkrController = {
  groupsAll: async (req, res, next) => {
    try {
      const year_id = parseInt(req.params.yearID); // Получаем id_year из параметра маршрута
      const years = await yearsModel.getById(year_id)
      const groups = await listsModel.groupsAll(year_id );
      res.render("result/groups", {
        title: 'Список групп',
        layout: 'layout2',
        groups,
        years
      })
    }
  }
};

```

```

    } catch (error){
      next(error);
    }
  },
  studentsAll: async (req, res, next) => {
    try {
      const group_id = parseInt(req.params.groupID); // Получаем id_year из параметра маршрута
      const year_id = parseInt(req.params.yearID); // Получаем id_year из параметра маршрута
      const resul = await listsModel.getAll4(group_id);
      const years = await yearsModel.getById(year_id)

      res.render('result/students', {
        title: 'Результаты',
        layout: 'layout2',
        resul,
        years
      })
    } catch (error){
      next(error);
    }
  },
  studentBy: async (req, res, next) => {
    try {
      const id = parseInt(req.params.studentID); // Получаем id_year из параметра маршрута
      const student_id = parseInt(req.params.studentID);
      const year_id = parseInt(req.params.yearID); // Получаем id_year из параметра маршрута
      const years = await yearsModel.getById(year_id)
      const result = await studentsModel.getAll(id);
      const users = await usersModel.getAll();
      const lists = await listsModel.getAll2(student_id);
      const resu = await listsModel.getAll3(student_id);
      res.render('result/student', {
        title: 'Результаты',
        layout: 'layout2',
        result,
        users,
        lists,
        resu,
        years
      })
    } catch (error){
      next(error);
    }
  }
}
}
controllers/studentsController.js
const studentsModel = require('../models/studentsModel');
const yearsModel = require("../models/yearsModel");
const usersModel = require("../models/usersModel");
const groupsModel = require("../models/groupsModel");

```

```

module.exports = studentsController = {
  studentAll: async (req, res, next) => {
    try {
      const students = await studentsModel.studentAll();
      res.render('admin/students/students', {
        title: 'Студенты',
        students
      })
    } catch (error){

```

```

        next(error);
    }
},
newStudent: async (req, res, next) => {
    const groups = await studentsModel.groupsAll()
    try {
        res.render('admin/students/add', {
            title: 'Добавление студента',
            groups
        })
    } catch (error) {
        next(error);
    }
},
create: async (req, res, next) => {
    try {
        const student = await studentsModel.create(req.body);
        res.render('admin/students/edit', {
            title: 'Добавление студента',
            student
        })
    } catch (error) {
        next(error);
    }
    res.redirect('/students');
},
getById: async (req, res, next) => {
    try {
        const students = await studentsModel.getById(req.params.id);
        const groups = await studentsModel.groupsAll()
        res.render('admin/students/edit', {
            title: 'Редактирование студента',
            students,
            groups
        })
    } catch (error) {
        next(error);
    }
},
update: async (req, res, next) => {
    try {
        const student = await studentsModel.update(req.params.id, req.body);
        res.render('admin/students/edit', {
            title: 'Редактирование студента',
            student
        })
        // res.send(user);
    } catch (error) {
        next(error);
    }
    res.redirect('/students');
},
delete: async (req, res, next) => {
    try {
        const student = await studentsModel.delete(req.params.id);
        res.render('admin/students/students', {
            title: 'Удаление студента',
            student
        })
    } catch (error) {
        next(error);
    }
}

```



```

    }
    res.redirect('/students');
  },

  };
controllers/usersController.js
const usersModel = require('../models/usersModel');
const studentsModel = require("../../models/studentsModel");
const yearsModel = require("../models/yearsModel");

module.exports = usersController = {
  getAll: async (req, res, next) => {
    try {
      const users = await usersModel.getAll();
      res.render('admin/users/users', {
        title: 'Пользователи',
        users
      })
    } catch (error){
      next(error);
    }
  },
  newUser: async (req, res, next) => {
    const roles = await usersModel.rolesALL();
    const years = await usersModel.yearsALL();

    try {
      res.render('admin/users/add', {
        title: 'Добавление пользователя',
        roles,
        years
      })
    } catch (error) {
      next(error);
    }
  },
  create: async (req, res, next) => {
    try {
      const user = await usersModel.create(req.body);
      res.render('admin/users/edit', {
        title: 'Добавление пользователя',
        user
      })
    } catch (error) {
      next(error);
    }
    res.redirect('/users');
  },
  getById: async (req, res, next) => {
    const roles = await usersModel.rolesALL();
    const years = await usersModel.yearsALL();
    try {
      const users = await usersModel.getById(req.params.id);
      res.render('admin/users/edit', {
        title: 'Редактирование пользователя',
        users,
        roles,
        years
      })
    } catch (error) {
      next(error);
    }
  }
}

```

```

    }
  },
  update: async (req, res, next) => {
    try {
      const user = await usersModel.update(req.params.id, req.body);
      res.render('admin/users/edit', {
        title: 'Редактирование пользователя',
        user
      })
      // res.send(user);
    } catch (error) {
      next(error);
    }
    res.redirect('/users');
  },
  delete: async (req, res, next) => {
    try {
      const user = await usersModel.delete(req.params.id);
      res.render('admin/users/users', {
        title: 'Удаление',
        user
      })
    } catch (error) {
      next(error);
    }
    res.redirect('/users');
  }
};

controllers/vkrController.js
const studentsModel = require("../models/studentsModel");
const groupsModel = require("../models/groupsModel");
const vkrsModel = require("../models/vkrsModel");
const yearsModel = require("../models/yearsModel");
const docsModel = require("../models/docsModel");
const fileUpload = require('../middleware/upload_vkrs');
const fileUpload1 = require('../middleware/upload_vkrs1');

const multer = require("multer");
// const translate = require('translate');
// import translate from "translate";

module.exports = vkrController = {
  groupsAll: async (req, res, next) => {
    try {
      const year_id = parseInt(req.params.yearID); // Получаем id_year из параметра маршрута
      const years = await yearsModel.getById(year_id)
      const groups = await vkrsModel.groupsAll(year_id);
      res.render("vkr/list-groups", {
        title: 'Список групп',
        layout: 'layout2',
        groups,
        years
      })
    } catch (error){
      next(error);
    }
  },
  studentsAll: async (req, res, next) => {
    try {
      const year_id = parseInt(req.params.yearID);

```

```

const group_id = parseInt(req.params.groupID);
const students = await vkrsModel.studentAll(group_id, year_id);
const years = await yearsModel.getById(year_id)
res.render('vkr/students', {
  title: 'Список студентов',
  layout: 'layout2',
  students,
  years
})
} catch (error){
  next(error);
}
},
getAll: async (req, res, next) => {
  try {
    const year_id = parseInt(req.params.yearID);
    const group_id = parseInt(req.params.groupID);
    const student_id = parseInt(req.params.studentID); // Получаем id_year из параметра
    const vkrs = await vkrsModel.getAll2(student_id, year_id, group_id);
    const years = await yearsModel.getById(year_id)

    res.render('vkr/vkr', {
      title: 'Выпускная квалификационная работа',
      layout: 'layout2',
      vkrs,
      years
    })
  } catch (error){
    next(error);
  }
},
vkrAll: async (req, res, next) => {
  try {
    const vkrs = await vkrsModel.vkrAll();
    res.render('admin/vkrs/vkrs', {
      title: 'Выпускная кваликационная работа',
      vkrs
    })
  } catch (error) {
    next(error);
  }
},
newVKR: async (req, res, next) => {
  const students = await vkrsModel.studentsAll()
  try {
    res.render('admin/vkrs/add', {
      title: 'Добавить ВКР',
      students
    })
  } catch (error) {
    next(error);
  }
},
uploadFile: function(req, res){
  const upload = multer({
    storage: fileUpload.files.storage(),
    allowedFile: fileUpload.files.allowedFile,
  }).single('file');

```

маршрута

```

upload(req, res, function (err) {
  console.log(req.file.filename);
  console.log(req.body.student_id);

  const file = Buffer.from(req.file.filename, 'latin1').toString('utf8');
  const student_id=req.body.student_id;
  vkrsModel.create({file,student_id}),(err, uploadeddata)=>
  {
    if(err) return res.send(err);
    console.log(uploadeddata);
  })

  if (err instanceof multer.MulterError) {
    res.send(err);
  } else if (err) {
    res.send(err);
  }else{
    res.render('admin/vkrs/add');
  }
  res.redirect('/vkr');
})

},
create: async (req, res, next) => {
  try {
    const vkr = await vkrsModel.create(req.body);
    res.render('admin/vkrs/add', {
      title: 'Добавить ВКР',
      vkr
    })
  } catch (error) {
    next(error);
  }
  res.redirect('/vkrs');
},
delete: async (req, res, next) => {
  try {
    const vkr = await vkrsModel.delete(req.params.id);
    res.render('admin/vkrs/vkrs', {
      title: 'Удаление ВКР',
      vkr
    })
  } catch (error) {
    next(error);
  }
  res.redirect('/vkr');
}
});
controllers/yearsController.js
const yearsModel = require('../models/yearsModel');

module.exports = yearsController = {

  getAll: async (req, res, next) => {
    try {
      const years = await yearsModel.getAll();
      res.render('admin/index', {
        title: 'Года',
        years
      })
    }
  }
}

```

```

        // res.send(years);
      } catch (error) {
        next(error);
      }
    },
    yearsAll: async (req, res, next) => {
      try {
        const years = await yearsModel.yearsAll();
        res.render('admin/years/years', {
          title: 'Года',
          years
        })
      } catch (error) {
        next(error);
      }
    },
    newYear: async (req, res, next) => {
      try {
        res.render('admin/years/add', {
          title: 'Добавить год'
        })
      } catch (error) {
        next(error);
      }
    },
    create: async (req, res, next) => {
      try {
        const year = await yearsModel.create(req.body);
        res.render('admin/years/add', {
          title: 'Добавить год',
          year
        })
      } catch (error) {
        next(error);
      }
      res.redirect('/years');
    },
    getById: async (req, res, next) => {
      try {
        const years = await yearsModel.getById(req.params.id);
        res.render('admin/years/edit', {
          title: 'Редактировать год',
          years
        })
      } catch (error) {
        next(error);
      }
    },
    update: async (req, res, next) => {
      try {
        const year = await yearsModel.update(req.params.id, req.body);
        res.render('admin/years/edit', {
          title: 'Редактировать год',
          year
        })
        // res.send(user);
      } catch (error) {
        next(error);
      }
    }
  }

```

```

        res.redirect('/years');
    },
    delete: async (req, res, next) => {
        try {
            const year = await yearsModel.delete(req.params.id);
            res.render('admin/years/years', {
                title: 'Удаление года',
                year
            })
        } catch (error) {
            next(error);
        }
        res.redirect('/years');
    }
};

middleware/upload_docs.js
var multer = require('multer');
module.exports.files={
    storage:function(){
        var storage = multer.diskStorage({
            destination: function (req, file, cb) {
                cb(null, 'public/uploads/docs')
            },
            filename: function (req, file, cb) {
                cb(null, file.originalname)
            }
        })

        return storage;
    },
    allowedFile:function(req, file, cb) {

        if (!file.originalname.match(/\.(pdf|doc)$/)) {
            req.fileValidationError = 'Only files are allowed!';
            return cb(new Error('Only files are allowed!'), false);
        }
        cb(null, true);
    }
}

middleware/upload_vkrs.js
var multer = require('multer');
module.exports.files={
    storage:function(){
        var storage = multer.diskStorage({
            destination: function (req, file, cb) {
                cb(null, 'public/uploads/vkrs')
            },
            filename: function (req, file, cb) {
                cb(null, file.originalname)
            }
        })

        return storage;
    },
    allowedFile:function(req, file, cb) {

        if (!file.originalname.match(/\.(pdf|doc)$/)) {
            req.fileValidationError = 'Only files are allowed!';
            return cb(new Error('Only files are allowed!'), false);
        }
    }
}

```

```

    }
    cb(null, true);
  },

  storage1: function() {
    var storage1 = multer.diskStorage({
      destination1: function (req, file, cb) {
        cb(null, 'public/uploads/vkrs')
      },
      filename1: function (req, file, cb) {
        cb(null, file.originalname)
      }
    })

    return storage1;
  },
  allowedFile1: function(req, file, cb) {

    if (!file.originalname.match(/\.(pdf|doc)$/)) {
      req.fileValidationError = 'Only files are allowed!';
      return cb(new Error('Only files are allowed!'), false);
    }
    cb(null, true);
  },
}
models/ docsModel.js
const knex = require('../db/db');

module.exports = docsModel = {
  getAll: async (year_id) => {
    const docs = await knex('docs').where('year_id', year_id);
    return docs;
  },
  getAll1: async (year_id) => {
    const docs = await knex
      .select("docs.year_id")
      .where('year_id', year_id);
    return docs;
  },
  docsAll: async () => {
    const docs = await knex
      .select("docs.id", "docs.file", "docs.year_id", "years.year_name")
      .from("docs")
      .join("years", "docs.year_id", "years.id");
    return docs;
  },
  yearsAll: async () => {
    const years = await knex('years');
    const docs = await knex
      .select("docs.id", "docs.file", "docs.year_id", "years.year_name")
      .from("docs")
      .join("years", "docs.year_id", "years.id");
    return years;
  },
  create: async (doc) => {
    const docs = await knex('docs').insert(doc);
    return docs;
  },
  delete: async (id) => {
    const docs = await knex("docs").where("id", id).delete();
    return docs;
  }
}

```

```

    },
  };
models/ groupsModel.js
const knex = require('../db/db');

module.exports = groupsModel = {
  groupAll:async (year_id) => {
    const groups = await knex('docs').where('year_id', year_id);
    return groups;
  },
  groupsAll: async () => {
    const groups = await knex
      .select("groups.id", "groups.group_name", "groups.year_id", "years.year_name")
      .from("groups")
      .join("years", "groups.year_id", "years.id");
    return groups;
  },
  yearsAll: async () => {
    const years = await knex('years');
    const groups = await knex
      .select("groups.id", "groups.group_name", "groups.year_id", "years.year_name")
      .from("groups")
      .join("years", "groups.year_id", "years.id");
    return years;
  },

  create: async (group) => {
    const groups = await knex("groups").insert(group);
    return groups;
  },
  getById: async (id) => {
    const group = await knex("groups")
      .where('id', id);
    // .select("groups.id", "groups.group_name", "groups.year_id", "years.year_name")
    // .from("groups")
    // .join("years", "groups.year_id", "years.id")
    // .where("groups.id", id);
    return group;
  },
  update: async (id, group) => {
    const groups = await knex("groups")
      .select("groups.id", "groups.group_name", "groups.year_id", "years.year_name")
      .from("groups")
      .join("years", "groups.year_id", "years.id")
      .where("groups.id", id).update(
        {
          group_name: group.group_name,
          year_id: group.year_id,
        }
      );
    return groups;
  },
  delete: async (id) => {
    const groups = await knex("groups").where("id", id).delete();
    return groups;
  },
};
models/ listsModel.js
const knex = require('../db/db');

module.exports = listsModel =

```



```

{
  studentAll: async (group_id, year_id) => {
    const students = await knex
      .select("students.id", "students.student_fio", "students.group_id", "students.student_gpa",
        "groups.year_id", "groups.group_name")
      .from("students")
      .join("groups", "students.group_id", "groups.id")
      .where('students.group_id', group_id)
      .where('groups.year_id', year_id);

    return students;
  },
  groupsAll: async (year_id) => {
    const docs = await knex('groups').where('year_id', year_id);
    return docs;
  },
  getAll: async (student_id) => {
    const lists = await knex('lists')
      .select ('user_id', 'student_id', 'g1', 'g2', 'g3_1', 'g3_2', 'g3_3', 'g4_1', 'g4_2', 'g4_3', 'g4_4', 'g4_5',
        'g4_6')
      .select( knex.raw(' round(((g1+g2+g3_1+g3_2+g3_3)/5),1) as avg_vkr'))
      .select( knex.raw('round(((g4_1+g4_2+g4_3+g4_4+g4_5+g4_6)/6),1) as avg_protect '))
      .select(
        knex.raw('round(((round(((g1+g2+g3_1+g3_2+g3_3)/5),1)+round(((g4_1+g4_2+g4_3+g4_4+g4_5+g4_6)/6),1
        ))/2),1) as avg_user'))
      .where('student_id', student_id);
    return lists;
  },
  getAll2: async (student_id) => {
    const lists = await knex('lists')
      .select ('user_id', 'student_id', 'g1', 'g2', 'g3_1', 'g3_2', 'g3_3', 'g4_1', 'g4_2', 'g4_3', 'g4_4', 'g4_5',
        'g4_6')
      .select( knex.raw(' round(((g1+g2+g3_1+g3_2+g3_3)/5),1) as avg_vkr'))
      .select( knex.raw('round(((g4_1+g4_2+g4_3+g4_4+g4_5+g4_6)/6),1) as avg_protect '))
      .select(
        knex.raw('round(((round(((g1+g2+g3_1+g3_2+g3_3)/5),1)+round(((g4_1+g4_2+g4_3+g4_4+g4_5+g4_6)/6),1
        ))/2),1) as avg_user'))
      .select(knex.raw('user_fio as users'))
      .join("users", "lists.user_id", "users.id")

      .where('student_id', student_id);

    return lists;
  },
  getAll3: async (student_id) => {
    const lists = knex()
      .select(knex.raw(' round(avg(x.avg_user),1) as res from(select
        student_id, round(((round(((g1+g2+g3_1+g3_2+g3_3)/5),1)+round(((g4_1+g4_2+g4_3+g4_4+g4_5+g4_6)/6),1
        ))/2),1) as avg_user from lists ) x '))

      .where('student_id', student_id);

    return lists;
  },
  getAll4: async (group_id) => {
    const lists = knex()

      .select(knex.raw(' y.id, round(((y.res+y.gpa)/2),1) as result, y.fio, y.group_id from (select x.id
        as id, round(avg(x.avg_user),1) as res, x.fio as fio, x.gpa as gpa, x.group_id as group_id from (select student_id
        as id, student_fio as fio, student_gpa as gpa, group_id as group_id,

```

```

round(((round(((g1+g2+g3_1+g3_2+g3_3)/5),1)+round(((g4_1+g4_2+g4_3+g4_4+g4_5+g4_6)/6),1))/2),1) as
avg_user from lists join students on lists.student_id=students.id ) x group by id,fio, gpa, group_id)y\n'))
    .where('group_id', group_id);

    return lists;
  },

  create: async (list) => {
    const lists = await knex("lists").insert(list);
    return lists;
  },
};
models/ studentsModel.js
const knex = require('../db/db');

module.exports = studentsModel = {
  studentsAll: async (group_id) => {
    const students = await knex('students').where('group_id',group_id);
    return students;
  },
  getAll: async (id) => {
    const student = await knex('students').where('id', id);
    return student;
  },
  studentAll: async () => {
    const students = await knex
      .select("students.id", "students.student_fio", "students.student_gpa", "groups.group_name")
      .from("students")
      .join("groups", "students.group_id", "groups.id");
    return students;
  },
  groupsAll: async () => {
    const groups = await knex("groups");
    const students = await knex
      .select("students.id", "students.student_fio", "students.student_gpa", "groups.group_name")
      .from("students")
      .join("groups", "students.group_id", "groups.id");
    return groups;
  },

  create: async (student) => {
    const students = await knex("students").insert(student);
    return students;
  },
  getById: async (id) => {
    const student = await knex("students").where("id", id);
    return student;
  },

  update: async (id, student) => {
    const students = await knex
      .select("students.id", "students.student_fio", "students.student_gpa", "groups.group_name")
      .from("students")
      .join("groups", "students.group_id", "groups.id")
      .where("id", id).update({
        student_fio: student.student_fio,
        student_gpa: student.student_gpa,
        group_id: student.group_id,
      });
    return students;
  }
};

```

```

    },
    delete: async (id) => {
        const students = await knex("students").where("id", id).delete();
        return students;
    },
};
models/ usersModel.js
const knex = require('../db/db');

module.exports = usersModel = {
    getAll: async () => {
        const users = await knex
            .select("users.id", "users.user_fio", "users.login", "users.password",
"users.email", "users.role_id", "users.year_id", "roles.role_name", "years.year_name")
            .from("users")
            .join("roles", "users.role_id", "roles.id")
            .join("years", "users.year_id", "years.id");
        return users;
    },
    rolesALL: async (req, res, next) => {
        const roles = await knex('roles');
        const years = await knex('years');
        const users = await knex .select("users.id", "users.user_fio", "users.login", "users.password",
"users.email", "users.role_id", "users.year_id", "roles.role_name", "years.year_name")
            .from("users")
            .join("roles", "users.role_id", "roles.id")
            .join("years", "users.year_id", "years.id");
        return roles;
    },
    yearsALL: async (req, res, next) => {
        const years = await knex('years');
        const users = await knex .select("users.id", "users.user_fio", "users.login", "users.password",
"users.email", "users.year_id", "years.year_name")
            .from("users")
            .join("years", "users.year_id", "years.id");
        return years;
    },
    create: async (user) => {
        const users = await knex("users").insert(user);
        return users;
    },
    getById: async (id) => {
        const user = await knex('users').where('id', id);
        return user;
    },
    update: async (id, user) => {
        const users = await knex.select("users.id", "users.user_fio", "users.login", "users.password",
"users.email", "users.role_id", "users.year_id", "roles.role_name", "years.year_name")
            .from("users")
            .join("roles", "users.role_id", "roles.id")
            .join("years", "users.year_id", "years.id")
            .where('id', id).update({
                user_fio: user.user_fio,
                email: user.email,
                login: user.login,
                password: user.password,
                role_id: user.role_id,
                year_id: user.year_id,
            });
        return users;
    }
};

```

```

    },
    delete: async (id) => {
      const users = await knex("users").where("id", id).delete();
      return users;
    },
  };
};
models/ vkrsModel.js
const knex = require('../db/db');

module.exports = vkrsModel = {
  getAll: async (student_id) => {
    const vkrs = await knex('vkrs').where('student_id', student_id);
    return vkrs;
  },
  getAll2: async (student_id, year_id, group_id) => {
    const vkrs = await knex
      .select("vkrs.id", "vkrs.file", "vkrs.file_pz", "vkrs.student_id", "students.student_fio",
        "students.group_id", "groups.year_id")
      .from("vkrs")
      .join("students", "vkrs.student_id", "students.id")
      .join("groups", "students.group_id", "groups.id")

      .where('students.group_id', group_id)
      .where('groups.year_id', year_id)
      .where('vkrs.student_id', student_id);
    return vkrs;
  },
  vkrAll: async () => {
    const vkrs = await knex
      .select("vkrs.id", "vkrs.file", "vkrs.file_pz", "vkrs.student_id", "students.student_fio")
      .from("vkrs")
      .join("students", "vkrs.student_id", "students.id");
    return vkrs;
  },
  studentAll: async (group_id, year_id) => {
    const students = await knex
      .select("students.id", "students.student_fio", "students.group_id", "students.student_gpa",
        "groups.year_id", "groups.group_name")
      .from("students")
      .join("groups", "students.group_id", "groups.id")
      .where('students.group_id', group_id)
      .where('groups.year_id', year_id);

    return students;
  },
  groupsAll: async (year_id) => {
    const docs = await knex('groups').where('year_id', year_id);
    return docs;
  },
  studentsAll: async () => {

    const students = await knex('students');
    const vkrs = await knex
      .select("vkrs.id", "vkrs.file", "vkrs.file_pz", "vkrs.student_id", "students.student_fio")
      .from("vkrs")
      .join("students", "vkrs.student_id", "students.id");
    return students;
  },
  create: async (vkr) => {
    const vkrs = await knex("vkrs").insert(vkr);
    return vkrs;
  }
};

```

```

    },
    delete: async (id) => {
      const vkrs = await knex("vkrs").where("id", id).delete();
      return vkrs;
    },
  };
models/ yearssModel.js

const knex = require('../db/db');

module.exports = yearsModel = {
  getAll: async () => {
    const years = await knex('years');
    return years;
  },
  yearsAll: async () => {
    const years = await knex('years');
    return years;
  },
  create: async (year) => {
    const years = await knex("years").insert(year);
    return years;
  },
  getById: async (id) => {
    const year = await knex('years').where('id', id);
    return year;
  },
  update: async (id, year) => {
    const years = await knex('years').where('id', id).update({
      year_name: year.year_name,
    });
    return years;
  },
  delete: async (id) => {
    const years = await knex("years").where("id", id).delete();
    return years;
  },
};

routes/about.js
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/about/', function(req, res) {
  res.render('about', {
    title: 'Contact',
    pageTestScript: '/qa/tests-about.js'
  });
});
module.exports = router;
routes/contact.js
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/contact', function(req, res, next) {
  res.render('contact', { title: 'Contact' });
});

```

```

module.exports = router;
routes/docsRoute.js
const express = require('express');
const router = express.Router();
const docsController = require("../controllers/docsController");
const yearsController = require("../controllers/yearsController");
const listsController = require("../controllers/listController");

// router.route("/docs").get(docsController.getAll)
//
// module.exports = router;

router.route('/:yearID/docs').get(docsController.getAll);
router.route('/docs').get(docsController.docsAll);
router.route('/docs/new').get(docsController.newDoc);
router.route('/docs/new').post(docsController.uploadFile);
router.route("/docs/:id").delete(docsController.delete);

module.exports = router;
routes/groupsRoute.js
const express = require('express');
const router = express.Router();
const groupsController = require("../controllers/groupsController");
const studentsController = require("../controllers/studentsController");
const yearsController = require("../controllers/yearsController");

router.route("/groups").get(groupsController.groupsAll)
router.route("/groups/new").get(groupsController.newGroup);
router.route("/groups").post(groupsController.create);
router.route("/groups/:id/edit").get(groupsController.getById);
router.route("/groups/:id").put(groupsController.update);
router.route("/groups/:id").delete(groupsController.delete);

module.exports = router;
routes/listsRoute.js
const express = require('express');
const router = express.Router();
const listController = require("../controllers/listController");
router.route('/:yearID/list-group').get(listController.groupsAll);
router.route('/:yearID/list-group/:groupID').get(listController.studentsAll);
router.route('/:yearID/list-group/:groupID/:studentID').get(listController.getAll);

module.exports = router;
routes/resultRoute.js
const express = require('express');
const router = express.Router();
const resultController = require("../controllers/resultController");
const listController = require("../controllers/listController");

router.route('/:yearID/group').get(resultController.groupsAll);
router.route('/:yearID/group/:groupID').get(resultController.studentsAll);
router.route('/:yearID/group/:groupID/:studentID').get(resultController.studentBy);

module.exports = router;
routes/sstudentsRoute.js
const express = require('express');
const router = express.Router();
const studentsController = require("../controllers/studentsController");
const yearsController = require("../controllers/yearsController");

router.route('/students').get(studentsController.studentAll);

```

```

router.route("/students/new").get(studentsController.newStudent);
router.route("/students").post(studentsController.create);
router.route("/students/:id/edit").get(studentsController.getById);
router.route("/students/:id").put(studentsController.update);
router.route("/students/:id").delete(studentsController.delete);
// router.route("/students/parser").get(studentsController.parserStudent);

module.exports = router;
routes/usersRoute.js
const express = require('express');
const router = express.Router();
const usersController = require("../controllers/usersController");
const studentsController = require("../controllers/studentsController");

router.route("/users/").get(usersController.getAll);
router.route("/users/new").get(usersController.newUser);
router.route("/users").post(usersController.create);
router.route("/users/:id/edit").get(usersController.getById);
router.route("/users/:id").put(usersController.update);
router.route("/users/:id").delete(usersController.delete);

module.exports = router;
routes/vkrRoute.js
const express = require('express');
const router = express.Router();
const vkrController = require("../controllers/vkrController");
const yearsController = require("../controllers/yearsController");
const docsController = require("../controllers/docsController");

router.route('/:yearID/list-groups').get(vkrController.groupsAll);
router.route('/:yearID/list-groups/:groupID').get(vkrController.studentsAll);
router.route('/:yearID/list-groups/:groupID/:studentID').get(vkrController.getAll);
router.route('/vkr').get(vkrController.vkrAll);
router.route("/vkr/new").get(vkrController.newVKR);
router.route("/vkr/new").post(vkrController.uploadFile);
router.route("/vkr/:id").delete(vkrController.delete);

module.exports = router;
routes/yearsRoute.js
const express = require('express');
const router = express.Router();
const yearsController = require("../controllers/yearsController");

router.route("/").get(yearsController.getAll)
router.route("/years").get(yearsController.yearsAll);
router.route("/years/new").get(yearsController.newYear);
router.route("/years").post(yearsController.create);
router.route("/years/:id/edit").get(yearsController.getById);
router.route("/years/:id").put(yearsController.update);
router.route("/years/:id").delete(yearsController.delete);

module.exports = router;
views/admin/docs/add.hbs
<h1>{{ title }}</h1>
<h1>Документация</h1>
<form class="form" action="/docs/new" method="post" enctype="multipart/form-data" >
<!-- enctype="multipart/form-data" -->
  <div class="form-group">
    <label for="name"> Год</label>
    <select name="year_id" id="year_id">

```

```

        {{#each years}}
        <option value="{{this.id}}">{{this.year_name}}</option>
        {{/each}}
    </select>
</div>
<div class="form-group">
    <label>File</label>
    <input type="file" name="file" id="file">
</div>
    <button type="submit">Добавить</button>
</form>
views/admin/docs/docs,hbs

<div class="pull-left">
    <h1>Список Документов</h1>
    <a class="btn btn-primary btn-sm" href="docs/new">Добавить</a>

</div>
<table class="table table-bordered table-striped">
    <thead>
    <tr>
        <th>id</th>
        <th>Название</th>
        <th>Год</th>
        <th>Удаление</th>
    </tr>
    </thead>
    {{#if docs}}
    {{#each docs}}
    <tr>
        <td>{{this.id}}</td>
        <td>{{this.file}}</td>
        <td>{{this.year_name}}</td>
        <td style="text-align: center">
            <button class="btn btn-primary btn-sm" type="submit"
name="remove_levels{{this.id}}" value="delete"><span class="fa fa-times"></span> Удалить</button>
        </td>
        <form action="/docs/{{id}}?_method=DELETE" method="post">
            <div id="confirm{{this.id}}" class="modal" tabindex="-1">
                <div class="modal-dialog">
                    <div class="modal-content">
                        <div class="modal-header">
                            <h5 class="modal-title">Удаление</h5>
                        </div>
                        <div class="modal-body">
                            <p>Вы точно хотите удалить?</p>
                        </div>
                        <div class="modal-footer">
                            <button type="submit" class="btn btn-primary btn-sm edit"><i class="fa fa-
trash-o" aria-hidden="true"></i>Удалить</button>
                            <button type="button" data-dismiss="modal" class="btn">Отмена</button>
                        </div>
                    </div>
                </div>
            </div>
        </form>
    </tr>
    {{/each}}
    {{/if}}
</table>
{{#each docs}}

```



```

<script>
    $('button[name="remove_levels{{{this.id}}}"]').on('click', function(e) {
        var $form = $(this).closest('form');
        e.preventDefault();
        $('#confirm{{{this.id}}}').modal({
            backdrop: 'static',
            keyboard: false
        })
        .on('click', '#delete', function(e) {
            $form.trigger('submit');
        })
        .on('click', '#cancel', function(e){
            e.preventDefault();
            $('#confirm{{{this.id}}}').modal.model('hide');
        });
    });
</script>
{{/each}}
views/admin/groups/add,hbs
<h1>{{title}}</h1>

<form class="form" action="/groups" method="post">
    <div class="form-group">
        <label for="name">Группа</label>
        <input type="text" name="group_name" class="form-control" id="group_name"
placeholder="Введите группу">
    </div>
    <div class="form-group">
        <label for="name">Год</label>
        <select name="year_id">
            {{#each years}}
                <option value="{{this.id}}">{{this.year_name}}</option>
            {{/each}}
        </select>
    </div>
    <a class="btn btn-danger btn-sm" href="/groups">Назад</a>
    <button type="submit" class="btn btn-primary btn-sm">Добавить</button>
</form>
views/admin/groups/edit,hbs
<h1>{{title}}</h1>
{{#each groups}}
    <form class="form" action="/groups/{{this.id}}?_method=PUT" method="post">
        <div class="form-group">
            <label for="name">Группа</label>
            <input type="text" value="{{this.group_name}}" name="group_name" class="form-control"
id="group_name" placeholder="Введите год">
        </div>
        {{/each}}
        <div class="form-group">
            <label for="name">Год</label>
            <select name="year_id">
                {{#each years}}
                    <option value="{{this.id}}" id="year_id">{{this.year_name}}</option>
                {{/each}}
            </select>
        </div>
        <a class="btn btn-danger btn-sm" href="/groups">Назад</a>
        <button type="submit" class="btn btn-primary btn-sm">Редактировать</button>
    </form>
views/admin/groups/groups,hbs
<div class="pull-left">

```

```
<h1>Список групп</h1>
<a class="btn btn-primary btn-sm" href="/groups/new">Добавить</a>

</div>
<table class="table table-bordered table-striped">
  <thead>
    <tr>
      <th>id</th>
      <th>Группа</th>
      <th>Год</th>
      <th>Действие</th>
    </tr>
  </thead>
  {{# if groups }}
    {{# each groups }}
      <tr>
        <td>{{ this.id }}</td>
        <td>{{this.group_name}}</td>
        <td>{{this.year_name}}</td>
        <td style="text-align: center">
          <a class="btn btn-primary btn-sm" href="/groups/{{ id }}/edit">Редактировать</a>
          <button class="btn btn-primary btn-sm" type="submit"
name="remove_levels{{this.id}}" value="delete"><span class="fa fa-times"></span> Удалить</button>
        </td>
      </tr>
    </td>
    <form action="/groups/{{ id }}?_method=DELETE" method="post">

    <div id="confirm{{this.id}}" class="modal" tabindex="-1">
      <div class="modal-dialog">
        <div class="modal-content">
          <div class="modal-header">
            <h5 class="modal-title">Удаление</h5>
          </div>
          <div class="modal-body">
            <p>Вы точно хотите удалить?</p>
          </div>
          <div class="modal-footer">
            <button type="submit" class="btn btn-primary btn-sm edit"><i class="fa fa-trash-
o" aria-hidden="true"></i>Удалить</button>
            <button type="button" data-dismiss="modal" class="btn">Отмена</button>
          </div>
        </div>
      </div>
    </div>
  </div>
  </form>
</tr>
  {{{/ each }}
  {{/ if }}
</table>

{{#each groups}}
<script>
$(button[name="remove_levels{{this.id}}"]).on('click', function(e) {
  var $form = $(this).closest('form');
  e.preventDefault();
  $('#confirm{{this.id}}').modal({
    backdrop: 'static',
    keyboard: false
  })
  .on('click', '#delete', function(e) {
    $form.trigger('submit');
  })
})
```

```

        .on('click', '#cancel', function(e){
            e.preventDefault();
            $('#confirm{{ this.id }}').modal.model('hide');
        });
    });
</script>
{{ /each }}
views/admin/students/add,hbs
<h1>{{ title }}</h1>

<form class="form" action="/students" method="post">
    <div class="form-group">
        <label for="name">ФИО</label>
        <input type="text" name="student_fio" class="form-control" id="student_fio"
placeholder="Введите ФИО">
    </div>
    <div class="form-group">
        <label for="name">Группа</label>
        <select name="group_id">
            {{#each groups}}
                <option value="{{ this.id }}">{{ this.group_name }}</option>
            {{/each}}
        </select>
    </div>
    <div class="form-group">
        <label for="name">Средняя оценка</label>
        <input type="text" name="student_gpa" class="form-control" id="student_gpa"
placeholder="Введите ср.оценку">
    </div>
    <a class="btn btn-danger btn-sm" href="/students">Назад</a>
    <button type="submit" class="btn btn-primary btn-sm">Добавить</button>
</form>
views/admin/ students /edit,hbs
<h1>{{ title }}</h1>

{{# each students }}
    <form class="form" action="/students/{{ this.id }}?_method=PUT" method="post">
        <div class="form-group">
            <label for="name">ФИО</label>
            <input type="text" value="{{ this.student_fio }}" name="student_fio" class="form-control"
id="student_fio" placeholder="Введите ФИО">
        </div>

        <div class="form-group">
            <label for="name">Средняя оценка</label>
            <input type="text" value="{{ this.student_gpa }}" name="student_gpa" class="form-control"
id="student_gpa" placeholder="Введите ср.оценку">
        </div>
    {{/each}}
    <div class="form-group">
        <label for="name">Год</label>
        <select name="group_id">
            {{#each groups}}
                <option value="{{ this.id }}" id="group_id">{{ this.group_name }}</option>
            {{/each}}
        </select>
    </div>
    <a class="btn btn-danger btn-sm" href="/students">Назад</a>
    <button type="submit" class="btn btn-primary btn-sm">Добавить</button>
</form>
views/admin/ students / students,hbs

```



```

        e.preventDefault();
        $('#confirm{{this.id}}').modal.model('hide');
    });
</script>
{{/each}}

views/admin/users/add,hbs
<h1>{{title}}</h1>

<form class="form" action="/users" method="post">
    <div class="form-group">
        <label for="name">ФИО</label>
        <input type="text" name="user_fio" class="form-control" id="user_fio" placeholder="Введите
ФИО">
    </div>
    <div class="form-group">
        <label for="name">Логин</label>
        <input type="text" name="login" class="form-control" id="login" placeholder="Введите
логин">
    </div>
    <div class="form-group">
        <label for="name">Пароль</label>
        <input type="text" name="password" class="form-control" id="password"
placeholder="Введите пароль">
    </div>
    <div class="form-group">
        <label for="name">Email</label>
        <input type="text" name="email" class="form-control" id="email" placeholder="Введите
email">
    </div>
    <div class="form-group">
        <label for="name">Год</label>
        <select name="year_id">
            {{#each years}}
                <option value="{{this.id}}">{{this.year_name}}</option>
            {{/each}}
        </select>
    </div>
    <div class="form-group">
        <label for="name">Роль</label>
        <select name="role_id">
            {{#each roles}}
                <option value="{{this.id}}">{{this.role_name}}</option>
            {{/each}}
        </select>
    </div>
    <a class="btn btn-danger btn-sm" href="/years">Назад</a>
    <button type="submit" class="btn btn-primary btn-sm">Добавить</button>
</form>

views/admin/users/edit,hbs
<h1>{{title}}</h1>
{{#each users}}
    <form class="form" action="/users/{{this.id}}?_method=PUT" method="post">
        <div class="form-group">
            <label for="name">ФИО</label>
            <input type="text" value="{{this.user_fio}}" name="user_fio" class="form-control"
id="user_fio" placeholder="Введите ФИО">
        </div>
        <div class="form-group">

```

```

        <label for="name">Логин</label>
        <input type="text" value="{{ this.login }}" name="login" class="form-control" id="login"
placeholder="Введите логин">
    </div>
    <div class="form-group">
        <label for="name">Пароль</label>
        <input type="text" value="{{ this.password }}" name="password" class="form-control"
id="password" placeholder="Введите пароль">
    </div>
    <div class="form-group">
        <label for="name">Email</label>
        <input type="text" value="{{ this.email }}" name="email" class="form-control" id="email"
placeholder="Введите email">
    </div>
    {{/each}}
    <div class="form-group">
        <label for="name">Год</label>
        <select name="year_id">
            {{#each years}}
                <option value="{{ this.id }}" id="year_id">{{ this.year_name }}</option>
            {{/each}}
        </select>
    </div>
    <div class="form-group">
        <label for="name">Роль</label>
        <select name="role_id">
            {{#each roles}}
                <option value="{{ this.id }}" id="role_id">{{ this.role_name }}</option>
            {{/each}}
        </select>
    </div>
    <a class="btn btn-danger btn-sm" href="/users">Назад</a>
    <button type="submit" class="btn btn-primary btn-sm">Добавить</button>
</form>
views/admin/ users / users,hbs
<div class="pull-left">
    <h1>Список Пользователей</h1>
    <a class="btn btn-primary btn-sm" href="users/new"> Добавить</a>
</div>
<table class="table table-bordered table-striped">
    <thead>
        <tr>
            <th>id</th>
            <th>ФИО</th>
            <th>Логин</th>
            <th>Пароль</th>
            <th>Email</th>
            <th>Год</th>
            <th>Роль</th>
            <th>Действие</th>
        </tr>
    </thead>
    {{# if users }}
    {{# each users }}
        <tr>
            <td>{{ this.id }}</td>
            <td>{{ this.user_fio }}</td>
            <td>{{ this.login }}</td>
            <td>{{ this.password }}</td>
            <td>{{ this.email }}</td>
            <td>{{ this.year_name }}</td>

```

```

        <td>{{ this.role_name }}</td>
        <td style="text-align: center">
            <a class="btn btn-primary btn-sm" href="/users/{{ id }}/edit"> Редактировать</a>
            <button class="btn btn-primary btn-sm" type="submit"
name="remove_levels{{ this.id }}" value="delete"><span class="fa fa-times"></span> Удалить</button>
        </td>
        <form action="/users/{{ id }}?_method=DELETE" method="post">
            <div id="confirm{{ this.id }}" class="modal" tabindex="-1">
                <div class="modal-dialog">
                    <div class="modal-content">
                        <div class="modal-header">
                            <h5 class="modal-title">Удаление</h5>
                        </div>
                        <div class="modal-body">
                            <p>Вы точно хотите удалить?</p>
                        </div>
                        <div class="modal-footer">
                            <button type="submit" class="btn btn-primary btn-sm edit"><i class="fa fa-trash-
o" aria-hidden="true"></i>Удалить</button>
                            <button type="button" data-dismiss="modal" class="btn">Отмена</button>
                        </div>
                    </div>
                </div>
            </div>
        </form>
    </tr>
    {{ /each }}
    {{ /if }}
</table>
{{ #each users }}
<script>
    $('button[name="remove_levels{{ this.id }}"]').on('click', function(e) {
        var $form = $(this).closest('form');
        e.preventDefault();
        $('#confirm{{ this.id }}').modal({
            backdrop: 'static',
            keyboard: false
        })
        .on('click', '#delete', function(e) {
            $form.trigger('submit');
        })
        .on('click', '#cancel', function(e){
            e.preventDefault();
            $('#confirm{{ this.id }}').modal.model('hide');
        });
    });
</script>
{{ /each }}
views/admin/vkrs/add,hbs
<h1>{{ title }}</h1>
<h1>Выпускная квалификационная работа</h1>
<form action="/vkr/new" method="POST" enctype="multipart/form-data">
    <div class="form-group">
        <label for="name"> Студент</label>
        <select name="student_id" id="student_id">
            {{ #each students }}
                <option value="{{ this.id }}">{{ this.student_fio }}</option>
            {{ /each }}
        </select>
    </div>
    <div class="form-group">

```

```

        <label>File</label>
        <input type="file" name="file" id="file">
    </div>
    <div class="form-group">
        <label>File_pz</label>
        <input type="file" name="file_pz" id="file_pz">
    </div>
    <button type="submit" > Добавить</button>
</form>
views/admin/vkrs/vkrs,hbs
<h1>{{ title }}</h1>
<div class="pull-left">
    <h1>Список ВКР</h1>
    <a class="btn btn-primary btn-sm" href="vkr/new"> Добавить</a>
</div>
<table class="table table-bordered table-striped">
    <thead>
        <tr>
            <th>id</th>
            <th>ФИО студента</th>
            <th>Пояснительная записка</th>
            <th>Техническое задание</th>
            <th>Удаление</th>
        </tr>
    </thead>
    {{# if vkrs }}
        {{# each vkrs }}
            <tr>
                <td>{{ this.id }}</td>
                <td>{{ this.student_fio }}</td>
                <td>{{ this.file }}</td>
                <td>{{ this.file_pz }}</td>
                <td style="text-align: center">
                    <button class="btn btn-primary btn-sm" type="submit"
name="remove_levels{{ this.id }}" value="delete"><span class="fa fa-times"></span> Удалить</button>
                </td>
                <form action="/vkr/{{ id }}?_method=DELETE" method="post">
                    <div id="confirm{{ this.id }}" class="modal" tabindex="-1">
                        <div class="modal-dialog">
                            <div class="modal-content">
                                <div class="modal-header">
                                    <h5 class="modal-title">Удаление</h5>
                                </div>
                                <div class="modal-body">
                                    <p>Вы точно хотите удалить?</p>
                                </div>
                                <div class="modal-footer">
                                    <button type="submit" class="btn btn-primary btn-sm edit"><i class="fa fa-
trash-o" aria-hidden="true"></i>Удалить</button>
                                    <button type="button" data-dismiss="modal" class="btn">Отмена</button>
                                </div>
                            </div>
                        </div>
                    </div>
                </form>
            </tr>
        {{/ each }}
    {{/ if }}
</table>
{{#each vkrs}}
<script>

```



```

$(button[name="remove_levels{{{this.id}}}"]).on('click', function(e) {
    var $form = $(this).closest('form');
    e.preventDefault();
    $('#confirm{{{this.id}}'}).modal({
        backdrop: 'static',
        keyboard: false
    })
    .on('click', '#delete', function(e) {
        $form.trigger('submit');
    })
    .on('click', '#cancel', function(e){
        e.preventDefault();
        $('#confirm{{{this.id}}'}).modal.model('hide');
    });
});
</script>
{{/each}}
views/admin/years/add,hbs
<h1>{{title}}</h1>
<form class="form" action="/years" method="post">
    <div class="form-group">
        <label for="name">Год</label>
        <input type="number" value="" name="year_name" class="form-control" id="year_name"
placeholder="Введите год">
    </div>
    <a class="btn btn-danger btn-sm" href="/years">Назад</a>
    <button type="submit" class="btn btn-primary btn-sm">Добавить</button>
</form>
views/admin/ years /edit,hbs
<h1>{{title}}</h1>
{{#each years}}
<form class="form" action="/years/{{this.id}}?_method=PUT" method="post">
    <div class="form-group">
        <label for="name">Год</label>
        <input type="number" value="{{this.year_name}}" name="year_name" class="form-control"
id="year_name" placeholder="Введите год">
    </div>
    <a class="btn btn-danger btn-sm" href="/years">Назад</a>
    <button type="submit" class="btn btn-primary btn-sm">Добавить</button>
</form>
views/admin/ years / years,hbs
<div class="pull-left">
    <h1>Список годов</h1>
    <a class="btn btn-primary btn-sm" href="years/new"> Добавить</a>
</div>
<table class="table table-bordered table-striped">
    <thead>
        <tr>
            <th>id</th>
            <th>Год</th>
            <th>Действие</th>
        </tr>
    </thead>
    {{# if years }}
        {{# each years }}
            <tr>
                <td>{{ this.id }}</td>
                <td>{{this.year_name}}</td>
                <td style="text-align: center">
                    <a class="btn btn-primary btn-sm" href="/years/{{ id }}/edit"> Редактировать</a>

```

```

        <button class="btn btn-primary btn-sm" type="submit" name="remove_levels{{ this.id
    }}" value="delete"><span class="fa fa-times"></span> Удалить</button>
    </td>
    <form action="/years/{{ id }}"?_method=DELETE" method="post">
        <div id="confirm{{ this.id }}" class="modal" tabindex="-1">
            <div class="modal-dialog">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 class="modal-title">Удаление</h5>
                    </div>
                    <div class="modal-body">
                        <p>Вы точно хотите удалить?</p>
                    </div>
                    <div class="modal-footer">
                        <button type="submit" class="btn btn-primary btn-sm edit"><i class="fa fa-trash-
o" aria-hidden="true"></i>Удалить</button>
                        <button type="button" data-dismiss="modal" class="btn">Отмена</button>
                    </div>
                </div>
            </div>
        </div>
    </div>
</form>
</tr>
{{ /each }}
{{ /if }}
</table>

{{ #each years }}
<script>
    $(button[name="remove_levels{{ this.id }}"]).on('click', function(e) {
        var $form = $(this).closest('form');
        e.preventDefault();
        $('#confirm{{ this.id }}').modal({
            backdrop: 'static',
            keyboard: false
        })
        .on('click', '#delete', function(e) {
            $form.trigger('submit');
        })
        .on('click', '#cancel', function(e){
            e.preventDefault();
            $('#confirm').modal.model('hide');
        });
    });
</script>
{{ /each }}
views/admin/index.js
<h1> Уважаемый секретарь ГЭК ИТИС! </h1>
<h3>Выберите нужный вам год для работы</h3>

<div class="container mt-4">

    {{ # each years }}
        <div><h3> <a class="text" href="/{{ this.id }}/docs"> {{ this.year_name }} </a></h3></div>
    {{ /each }}
views/list/groups.hbs
<h1> {{ title }} </h1>
<div class="container mt-4">
    {{ # each groups }}
        <div><h3> <a href="{{ this.id }}"> {{ this.group_name }} </a></h3></div>
    {{ /each }}

```



```

</tr>
<tr>
  <td rowspan="3">Результаты</td>
  <td>1. Наличие практических рекомендаций по решению поставленных в работе задач,
наличие методик использования программного продукта для программиста и пользователя</td>
  <td>5</td>
  <td>
    <div class="form-group">
      {{this.g3_1}}
    </div>
  </td>
</tr>
<tr>
  <td>2. Достоверность и обоснованность выводов по
    проведенному исследованию, их соответствие
    заявленной цели</td>
  <td>5</td>
  <td><div class="form-group">
    {{this.g3_2}}
  </div></td> </tr>
<tr>
  <td>3. Апробация результатов исследования (доклады на
    научном семинаре или конференции, публикации,
    рекомендации к внедрению и др.)</td>
  <td>5</td>
  <td><div class="form-group">
    {{this.g3_3}}
  </div></td> </tr>
<tr>
  <td>1</td>
  <td>Средний балл за ВКР</td>
  <td>5</td>
  <td><div class="form-group">
    {{this.avg_vkr}}
  </div>
</td>
</tr>
<tr>
  <td rowspan="6">Защита ВКР</td>
  <td>1. Степень структурированности и логичности доклада</td>
  <td>5</td>
  <td><div class="form-group">
    {{this.g4_1}}
  </div></td>
</tr>
<tr>
  <td>2. Использование демонстрационного материала, его
    презентабельность (наличие презентации)</td>
  <td>5</td>
  <td><div class="form-group">
    {{this.g4_2}}
  </div></td>
</tr>
<tr>
  <td>3. Научная аргументация и защита своей точки зрения,
    навыки публичной речи</td>
  <td>5</td>
  <td><div class="form-group">
    {{this.g4_3}}
  </div></td>

```

```

</tr>
<tr>
  <td>4. Четкость и аргументированность выводов по
    результатам исследования</td>
  <td>5</td>
  <td><div class="form-group">
    {{this.g4_4}}
  </div></td>
</tr>
<tr>
  <td>5. Четкость и аргументированность позиции студента при
    ответе на вопросы членов ГЭК, на замечания
    руководителя и рецензента</td>
  <td>5</td>
  <td><div class="form-group">
    {{this.g4_5}}
  </div></td>
</tr>
<tr>
  <td>6. Общий уровень общения с аудиторией</td>
  <td>5</td>
  <td><div class="form-group">
    {{this.g4_6}}
  </div></td>
</tr>
<tr>
  <td>2</td>
  <td>Средний балл за защиту ВКР</td>
  <td>5</td>
  <td><div class="form-group">
    {{this.avg_protect}}
  </div></td>
</tr>
</tbody>
</table>
</form>
{{/each}}
</div>
</div>
views/list/new.hbs
<h1> {{title}} </h1>
<div class="card">
  <div class="card-header">
    <div class="row">
      <table class="table table-bordered">
        <thead>
          <tr>
            <th scope="col">Наименование</th>
            <th scope="col">Показатели освоения компетенции согласно ФГОС и ОПОП</th>
            <th scope="col">Максимальный балл</th>
            <th scope="col">Оцениваемый балл</th>
          </tr>
        </thead>
        {{#each lists}}
          <form id="add_form" method="post">
            <tbody>
              <tr>
                <td rowspan="3">Постановка цели и задач исследования</td>
                <td>Актуальность и целевая направленность работы</td>

```

```
  |
```

```

        научном семинаре или конференции, публикации,
        рекомендации к внедрению и др.)</td>
<td>5</td>
<td><div class="form-group">
    <input type="number" value="{{this.g3_3}}" name="g3_3" class="form-control" id="g3_3"
placeholder="балл">
</div></td> </tr>
<tr>
<td>1</td>
<td>Средний балл за ВКР</td>
<td>5</td>
<td><div class="form-group">
    <input type="number" value="{{this.avg_vkr}}" name="avg_vkr" class="form-control"
id="avg_vkr" placeholder="балл">
</div>
</td>
</tr>
<tr>
<td rowspan="6">Защита ВКР</td>
<td>1.Степень структурированности и логичности доклада</td>
<td>5</td>
<td><div class="form-group">
    <input type="number" value="{{this.g4_1}}" name="g4_1" class="form-control" id="g4_1"
placeholder="балл">
</div></td>
</tr>
<tr>
<td>2.Использование демонстрационного материала, его
презентабельность (наличие презентации)</td>
<td>5</td>
<td>
<div class="form-group">
    <input type="number" value="{{this.g4_2}}" name="g4_2" class="form-control"
id="g4_2" placeholder="балл">
</div></td>
</tr>
<tr>
<td>3.Научная аргументация и защита своей точки зрения,
навыки публичной речи</td>
<td>5</td>
<td><div class="form-group">
    <input type="number" value="{{this.g4_3}}" name="g4_3" class="form-control" id="g4_3"
placeholder="балл">
</div></td>
</tr>
<tr>
<td>4.Четкость и аргументированность выводов по
результатам исследования</td>
<td>5</td>
<td><div class="form-group">
    <input type="number" value="{{this.g4_4}}" name="g4_4" class="form-control" id="g4_4"
placeholder="балл">
</div></td>
</tr>
<tr>
<td>5.Четкость и аргументированность позиции студента при
ответе на вопросы членов ГЭК, на замечания
руководителя и рецензента</td>
<td>5</td>
<td><div class="form-group">

```

```

        <input type="number" value="{{ this.g4_5 }}" name="g4_5" class="form-control" id="g4_5"
placeholder="балл">
    </div></td>
</tr>
<tr>
    <td>6.Общий уровень общения с аудиторией</td>
    <td >5</td>
    <td ><div class="form-group">
        <input type="number" value="{{ this.g4_6 }}" name="g4_6" class="form-control" id="g4_6"
placeholder="балл">
    </div></td>
</tr>
<tr>
    <td>2</td>
    <td>Средний балл за защиту ВКР</td>
    <td >5</td>
    <td ><div class="form-group">
        <input type="number" step=".01" value="{{ this.avg_protect }}" name="avg_protect"
class="form-control" id="avg_protect" placeholder="балл">
    </div></td>
</tr>
</tbody>
</table>
</form>
    {{ /each }}
<button form="add_form" type="submit" class="btn btn-primary mt-4">Заполнить</button>
</div>
</div>
views/list/students.hbs
<h1> {{ title }} </h1>
<div class="container mt-4">
    {{# each students }}
        <div><h3> <a href="{{ this.group_id }}" /> {{ this.id }} > {{ this.student_fio }} </a></h3></div>
    {{ /each }}
</div>

views/partials/admin_menu.hbs
<div class="container-fluid" style="margin-bottom: 100px">
    <div class="row">
        <div class="col-md-2">
            <h3> Меню</h3>
            <table class="table table-bordered table-striped">
                <tr>
                    <td style="padding-left: 30px">
                        <i class="fa fa-angle-double-right"></i> <a class="text" href="/">Главная</a>
                    </td>
                </tr>
                <tr>
                    <td style="padding-left: 30px">
                        <i class="fa fa-angle-double-right"></i> <a class="text" href="/years">Года</a>
                    </td>
                </tr>
                <tr>
                    <td style="padding-left: 30px">
                        <i class="fa fa-angle-double-right"></i> <a class="text"
href="/users">Пользователи</a>
                    </td>
                </tr>
                <tr>
                    <td style="padding-left: 30px">

```



```

        <i class="fa fa-angle-double-right"></i> <a class="text" href="/groups">Группы</a>
    </td>
</tr>
<tr>
    <td style="padding-left: 30px">
        <i class="fa fa-angle-double-right"></i> <a class="text" href="/students">Студенты</a>
    </td>
</tr>
<tr>
    <td style="padding-left: 30px">
        <i class="fa fa-angle-double-right"></i> <a class="text" href="/vkr">ВКР</a>
    </td>
</tr>
<tr>
    <td style="padding-left: 30px">
        <i class="fa fa-angle-double-right"></i> <a class="text" href="/docs">Приказы</a>
    </td>
</tr>
</table>
</div>
views/partials/menu.hbs
<header class="site-header">
    <nav class="navbar navbar-expand-md navbar-dark " style="background-color: #89c1ec;">
        <div class="container">
            <a class="navbar-brand mr-4" href="#"> Диплом ИТИС</a>
            <div class="collapse navbar-collapse" id="navbarToggle">
                <div class="navbar-nav mr-auto">
                    {{#each years}}
                    <a class="nav-item nav-link " href="/{{this.id}}/docs/">Приказы</a>
                    <a class="nav-item nav-link " href="/{{this.id}}/list-groups/">ВКР</a>
                    <a class="nav-item nav-link" href="/{{this.id}}/list-group/">Оценочные листы</a>
                    {{/each}}
                </div>
            </div>
        </div>
    </nav>
</header>
views/result/groups.hbs
<h1> {{title}} </h1>
<div class="container mt-4">
    {{#each groups}}
        <div><h3> <a href="{{this.id}}"> {{this.group_name}} </a></h3></div>
    {{/each}}
</div>
views/result/student.hbs

<!--<h1> {{title}} </h1>-->
<div class="container">
    <span id="message"></span>
    <div class="card">
        <div class="card-body">
            <h1 class="mt-4 mb-4 text-center text-primary"><b>{{title}} </b></h1>
            <div class="table-responsive">
                {{#each result}}
                    {{this.student_fio}}
                {{/each}}
            <table class="table table-striped table-bordered" id="sample_data">
                <thead>
                    <tr>
                        <th>ФИО проверяющего</th>

```

```

        <th>Средний балл</th>
    </tr>
</thead>
<tbody> {{#each lists}}
    <tr>
        <td>{{this.users}}</td>
        <td>{{this.avg_user}}</td>

    </tr>
    {{/each}}
    {{#each resu}}
    <tr>
        <td>Итоговая оценка</td>
        <td>{{this.res}}</td>
    </tr>
    {{/each}}
</tbody>
</table>
</div>
</div>
</div>
views/result/students.hbs

<!--<h1> {{title}} </h1>-->
<div class="container">
    <span id="message"></span>
    <div class="card">
        <div class="card-body">
            <h1 class="mt-4 mb-4 text-center text-primary"><b>{{title}} </b></h1>
            <div class="table-responsive">
                <table class="table table-striped table-bordered" id="sample_data">
                    <thead>
                        <tr>
                            <th>id</th>
                            <th>ФИО</th>
                            <th>Итоговая оценка</th>
                        </tr>
                    </thead>
                    <tbody>{{# each resul}}
                        <tr>
                            <td>{{this.id}}</td>
                            <td><a href="{{this.group_id}}/{{this.id}}">{{this.fio}}</a></td>
                            <td>{{this.result}}</td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</div>
views/vkr/list-groups.hbs
<h1> {{title}} </h1>
    {{# each groups}}
        <div><h3> <a href="{{this.id}}"> {{this.group_name}} </a></h3></div>
    {{/each}}
views/ vkr /students.hbs
<h1> Уважаемые члены ГЭК ИТИС! </h1>
<div class="container mt-4">
    <h1> {{title}} </h1>

```

```

    {{# each students}}
    <div><h3> <a href="{{this.group_id}}/{{this.id}}"> {{this.student_fio}} </a></h3></div>
    {{/each}}
</div>

views/ vkr /vkr.hbs

<h1> {{title}} </h1>
{{# each vkrs}}
<!-- <h3><a href="/uploads/vkrs/{{this.file}}"> открыть {{this.file}} </a></h3>-->
<h3><a href="#" download="{{this.file}}"> {{this.file}} <p>скачать</p></a></h3>
<h3><a href="#" download="{{this.file1}}"> {{this.file_pz}} <p>скачать</p></a></h3>
<!-- <h3> <a href=""> {{this.vkr_pz}} </a></h3>-->
{{/each}}
views/about.hbs
<p>О проекте...</p>
<a href="/contact">Контакты</a>
views/contact.hbs
<h1>{{title}}</h1>
<p>Email:luba260902@gmail.com</p>
<p>GitHub:https://github.com/Lubov2609/Diplom</p>
views/docs.hbs
<h1>{{title}}</h1>
<!--{{# each docs}}-->
<!-- <div><h3> <a href="/uploads/docs/{{this.file}}" >{{this.file}}<p>открыть файл в
браузере</p></a> </h3></div>-->
<!--{{/each}}-->
{{# each docs}}
<div><h3> <a download="{{this.file}}" href="#" >{{this.file}} <p>скачать файл<p> </a>
</h3></div>
{{/each}}
views/error.hbs
<h1>{{message}}</h1>
<h2>{{error.status}}</h2>
<pre>{{error.stack}}</pre>
views/layout.hbs
<html>
<style>
  body {
    background-color: #CDE6F8!important;
  }
</style>
<head>
  <!-- Обязательные мета-теги -->
  <meta charset="utf-8">
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOMLAsjC"
crossorigin="anonymous">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
  <!-- <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">-->
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
  <link href="https://cdn.datatables.net/1.12.0/css/dataTables.bootstrap5.min.css" rel="stylesheet">
  <script src="https://cdn.datatables.net/1.12.0/js/jquery.dataTables.min.js"></script>
  <script src="https://cdn.datatables.net/1.12.0/js/dataTables.bootstrap5.min.js"></script>
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

```

```

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
<link
href='http://fonts.googleapis.com/css?family=Roboto:400,100,100italic,300,300italic,400italic,500,500italic,700,700italic,900italic,900' rel='stylesheet' type='text/css'>
<!-- <link rel="stylesheet" type="text/css" href="{% static 'app/main.css' %}">-->
<link href='https://unpkg.com/boxicons@2.1.2/css/boxicons.min.css' rel='stylesheet'>
<title>Система автоматизации работы членов ГЭК/{{ title }}</title>
<link rel='stylesheet' href='stylesheets/style.css'/>
<link rel="stylesheet" href="/stylesheets/admin.css">
{{ #if showTests }}
<link rel="stylesheet" href='https://cdnjs.cloudflare.com/ajax/libs/mocha/2.1.0/mocha.css'>
{{ /if }}
<!-- <script src="//code.jquery.com/jquery-2.0.2.min.js"></script>-->
</head>
<body>
<header class="site-header" style="background-color: #5ea9e3">
<nav class="navbar navbar-inverse navbar-expand-md navbar-dark fixed-top " >
<div class="container">
<!-- <a class="navbar-brand mr-4" href="#"> Диплом ИТИС</a>-->
<div class="collapse navbar-collapse" id="navbarToggle">
<div class="navbar-nav mr-auto">
<p class="navbar-brand-name">Панель администрирования: Автоматизированная
система для работы ГЭК</p>
</div>
</div>
</div>
</nav>
</header>
{{> admin_menu}}
<div class="col-md-10">
{{{ body }}}
</div>
</div>
<footer class="page-footer" style="background-color: #5ea9e3">
<div class=" navbar navbar-fixed-bottom">
<div class="pull-right">
<h7 style="color: white">Диплом ИТИС </h7>
<a class=" text navbar-brand webstudio" href="/contact" >Разработка - ©
luba260902@gmail.com</a>
</div>
</div>
</div>
</div>
</div>
</div>
<!-- Дополнительные скрипты JavaScript -->
<!-- Сначала jQuery, затем Popper.js, последним Bootstrap JS -->

{{ #if showTests }}
<div id="mocha"></div>
<script src="https://cdnjs.cloudflare.com/ajax/libs/mocha/2.1.0/mocha.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/chai/2.0.0/chai.js"></script>
<script
type="text/javascript"
src="//cdnjs.cloudflare.com/ajax/libs/lodash.js/0.10.0/lodash.min.js"></script>
<!-- настраиваем Mocha: предстоит TDD-тестирование -->
<script>
mocha.ui('tdd');
```

```

        const assert = chai.assert;
    </script>
    <script src='/qa/global-tests.js'></script>
    {{#if pageTestScript}}
        <script src='{{pageTestScript}}'></script>
    {{/if}}
    <script>mocha.run();</script>
{{/if}}
</body>
</html>
<script>
if( $(document).height() <= $(window).height() ){
$('.page-footer').addClass("fixed-bottom");
}
</script>
views/ layout2.hbs

<!DOCTYPE html>
<html>
<style>
    body {
        background-color: #CDE6F8!important;
    }
</style>
<head>
    <!-- Обязательные мета-теги -->
    <meta charset="utf-8">
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- Bootstrap CSS -->
    <link
        href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfSpd3yD65VohhpuaCOMLASjC"
integrity="sha384-
crossorigin="anonymous">
    <link href="https://cdn.datatables.net/1.12.0/css/dataTables.bootstrap5.min.css" rel="stylesheet">
    <script
        src="https://code.jquery.com/jquery-3.6.0.js"
        integrity="sha256-
H+K7U5CnX11h5ywQfKtSj8PCmoN9aaq30gDh27Xc0jk=" crossorigin="anonymous"></script>
    <script
        src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>
    <script src="https://cdn.datatables.net/1.12.0/js/jquery.dataTables.min.js"></script>
    <script src="https://cdn.datatables.net/1.12.0/js/dataTables.bootstrap5.min.js"></script>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link
        href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
        rel="stylesheet"
        href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
        integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
        crossorigin="anonymous">
    <link
        href='http://fonts.googleapis.com/css?family=Roboto:400,100,100italic,300,300italic,400italic,500,500italic,700,700italic,900italic,900' rel='stylesheet' type='text/css'>
    <link rel="stylesheet" type="text/css" href="{% static 'app/main.css' %}">
    <link href="https://unpkg.com/boxicons@2.1.2/css/boxicons.min.css" rel="stylesheet">
    <title>Система автоматизации работы членов ГЭК/{{ title }}</title>
    <link rel="stylesheet" href="stylesheets/style.css">
    <link rel="stylesheet" href="/stylesheets/admin.css">
    {{#if showTests}}
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/mocha/2.1.0/mocha.css">
    {{/if}}
    <script src="//code.jquery.com/jquery-2.0.2.min.js"></script>
</head>

```

```

<body>
  {{> menu }}

  {{{body}}}
  <!-- Дополнительные скрипты JavaScript -->
  <!-- Сначала jQuery, затем Popper.js, последним Bootstrap JS -->
  <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2Dk1tkvYIK3UENzmM7KcRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYI"
crossorigin="anonymous"></script>
  {{#if showTests}}
    <div id="mocha"></div>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/mocha/2.1.0/mocha.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/chai/2.0.0/chai.js"></script>
    <script type="text/javascript"
src="//cdnjs.cloudflare.com/ajax/libs/lodash.js/0.10.0/lodash.min.js"></script>
    <!-- настраиваем Mocha: предстоит TDD-тестирование -->
    <script>
      mocha.ui('tdd');
      const assert = chai.assert;
    </script>
    <script src='/qa/global-tests.js'></script>
    {{#if pageTestScript}}
      <script src='{ {pageTestScript} }'></script>
    {{/if}}
    <script>mocha.run();</script>
  {{/if}}
<footer class="page-footer" style="background-color: #5ea9e3">
  <div class="navbar navbar-fixed-bottom">
    <div class="pull-right">
      <h7 style="color: white">Диплом ИТИС </h7>
      <a class="text navbar-brand webstudio" href="/contact">Разработка - ©
luba260902@gmail.com</a>
    </div>
  </div>
</footer>
</body>
</html>
<script>
  if( $(document).height() <= $(window).height() ){
    $(".page-footer").addClass("fixed-bottom");
  }
</script>
app.js

```

// Подключение библиотек для сайта

```

const createError = require('http-errors');
const express = require('express');
const path = require('path');
const parser = require('body-parser');
const methodOverride = require('method-override');
const cookieParser = require('cookie-parser');
const logger = require('morgan');
const config = require("config");
const upload = require('express-fileupload')

```

```

const hbs = require('hbs');

// Регистрация роутов для сайта
const aboutRouter = require('./routes/about');
const contactRouter = require('./routes/contact');
const yearsRouter = require('./routes/yearsRoute');
const docsRouter = require('./routes/docsRoute');
const usersRouter = require('./routes/usersRoute');
const studentsRouter = require('./routes/studentsRoute');
const groupsRouter = require('./routes/groupsRoute');
const vkrRouter = require('./routes/vkrRoute');
const listRouter = require('./routes/listRoute');
const resultRouter = require('./routes/resultRoute');

var app = express();

// Настройка шаблонов view
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'hbs');
hbs.registerPartials(__dirname + '/views/partials');
// app.set("trust proxy", true);
app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));
app.use(methodOverride('_method'));
// app.use(parser.urlencoded({extended: true}))
// app.use(parser.json())

//создание пути тестирования
app.use(function(req, res, next){
  res.locals.showTests = app.get('env') !== 'production' &&
    req.query.test === '1';
  next();
});
// Подключение роутов для проекта
app.use('/', usersRouter);
app.use('/', contactRouter);
app.use('/', docsRouter);
app.use('/', yearsRouter);
app.use('/', aboutRouter);
app.use('/', groupsRouter);
app.use('/', studentsRouter);
app.use('/', vkrRouter);
app.use('/', listRouter);
app.use('/', resultRouter);
// catch 404 and forward to error handler
app.use(function(req, res, next) {
  next(createError(404).send("Not Found"));
});
// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};
  // render the error page
  res.status(err.status || 500);
  res.render('error');
});
app.listen(process.env.PORT || config.port, () => {

```

```

    global.console.log(`Server is up and running on port ${config.port}`);
  });
  module.exports = app;
  knexfile.js
  require('dotenv').config()
  const pg = require('pg')
  pg.defaults.ssl = false

  module.exports = {
    development: {
      client: "postgresql",
      connection: {
        host: "localhost",
        user: "luba",
        password: "admin",
        database: "my_db"
      }
    },
    production: {
      client: "postgresql",
      connection: {
        host: "127.0.0.1",
        port: 5432,
        user: "postgres",
        password: "admin",
        database: "my_db"
      }
    }
  };

  package.json
  {
    "name": "diplom",
    "version": "1.0.0",
    "private": true,
    "scripts": {
      "start": "node ./bin/www",
      "dev": "nodemon --exec node app.js",
      "pm2": "pm2 kill && pm2 start --name skladnikluba.ru pm2.json && pm2 save && pm2 startup &&
pm2 save"
    },
    "dependencies": {
      "ajax": "^0.0.4",
      "bcrypt": "^5.1.1",
      "bcryptjs": "^2.4.3",
      "body-parser": "^1.20.2",
      "config": "^3.3.11",
      "cookie-parser": "~1.4.4",
      "cors": "^2.8.5",
      "crypto": "^1.0.1",
      "crypto-js": "^4.2.0",
      "csurf": "^1.11.0",
      "debug": "~2.6.9",
      "dotenv": "^16.4.5",
      "express": "^4.19.2",
      "express-fileupload": "^1.5.0",
      "express-validator": "^7.1.0",
      "hbs": "^4.2.0",
      "hex": "^0.1.0",
      "http": "^0.0.1-security",
      "http-errors": "~1.6.3",

```



```

    "jsonwebtoken": "^9.0.2",
    "knex": "^3.1.0",
    "method-override": "^3.0.0",
    "morgan": "~1.9.1",
    "multer": "*",
    "node.js": "*",
    "nodemon": "^3.1.0",
    "objection": "^3.1.4",
    "passport": "^0.7.0",
    "path": "^0.12.7",
    "pg": "^8.11.5",
    "pg-hstore": "^2.3.4",
    "puppeteer": "^22.10.0",
    "puppeteer-core": "^22.9.0",
    "sequelize": "^6.37.3",
    "serve-static": "^1.15.0",
    "slug": "^9.0.0",
    "translate": "^3.0.0"
  },
  "engines": {
    "node": "20.13.1",
    "npm": "10.5.2"
  },
  "devDependencies": {
    "chai": "^5.1.0",
    "fortune": "^5.5.19",
    "mocha": "^10.4.0"
  },
  "description": "$ npm install\r $ npm start",
  "main": "app.js",
  "repository": {
    "type": "git",
    "url": "git+https://github.com/Lubov2609/Diplom.git"
  },
  "author": "",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/Lubov2609/Diplom/issues"
  },
  "homepage": "https://github.com/Lubov2609/Diplom#readme"
}
package-lock.json

```

```

pm2.json
{
  "apps": [
    {
      "name": "skladnikluba.ru",
      "script": "app.js",
      "env": {
        "NODE_ENV": "production"
      },
      "log_date_format": "MM-DD-YYYY HH:mm Z",
      "error_file": "log/pm2_error.log",
      "out_file": "log/pm2_out.log"
    }
  ]
}

```